FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Gesture Recognition for Human-Robot Collaborative Assembly

Pedro Miguel Melo

Integrated Master in Informatics and Computing Engineering

Supervisor: Armando Sousa (Ph.D)

Co-Supervisor: Carlos Costa

July 19, 2018

Gesture Recognition for Human-Robot Collaborative Assembly

Pedro Miguel Melo

Integrated Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Luís Almeida (Ph.D)

External Examiner: Artur Pereira (Ph.D)

Supervisor: Armando Sousa (Ph.D)

July 19, 2018

Abstract

Human-robot collaboration plays an important role in the industry's future where robots and human operators work together in industrial scenarios. In order to properly cooperate with human workers, robots must be able to communicate in a way that seems natural for humans by means of an active perception of their surrounding environment. By recognizing and interpreting distinct gestures performed by their human co-workers during assembly stages, robots could cooperate by understanding and carrying out given commands.

Solutions for gesture recognition can be obtained combining not only computer vision but also data fusion and machine learning techniques. In order to be robust such solutions must cope with the sensors' limitations, the high number of *DoF* in the human hand and with the distinct degrees of occlusions from which they can suffer while handling distinct pieces.

This thesis consists in a solution for live recognition of both static and dynamic gestures targeting cooperative assembly workstations. The data was obtained using both the *Senso Gloves* and the *Leap Motion*. Five distinct hand features were considered for the recognition.

Static gesture recognition is performed using *Support Vector Machines*. Although state-of-the-art methodologies for hand pose estimation use convolutional neural networks, the developed system is able to recognize static gestures with minimal accuracy levels of 80%.

Due to their time dependant nature, dynamic gestures are modeled using *Hidden Markov Models*, decision motivated by the existing similarities between gestural communication and oral speech. To more efficiently train the models the *k-Means* clusterer algorithm is used. A cluster optimizer based on the *Silhouette* value is also used to improve the data clusterization. The developed system is able to recognize over five distinct gestures with minimal accuracy levels of 75%.

The performance of the presented solution was evaluated using cross-validation techniques applied over a personal gesture dataset consisting of collaborative assembly operations, allowing a robot to respond to commands and thus cooperate with humans during workpiece assembly by means of gestural communication.

Keywords

robotics, computer vision, data fusion, static gesture, dynamic gesture, gesture recognition, human-robot collaboration, machine learning

Resumo

A colaboração humano-robô é central para o futuro da indústria, no qual robôs e operadores humanos trabalham conjuntamente em cenários industriais. Por forma a conseguirem cooperar com seres humanos, os robôs devem ser capazes de comunicar com os mesmos numa forma o mais humana possível, através de uma perceção ativa do ambiente circundante. Ao serem capazes de reconhecer distintos gestos manuais executados pelos operários humanos durante a montagem de peças, os robôs poderiam cooperar através da interpretação e realização de comandos dados pelo operador.

Soluções para o reconhecimento de gestos podem ser obtidas combinando técnicas de visão por computador, de fusão de dados e de *machine learning*. Por forma a serem robustas, tais soluções precisam ainda de lidar com as limitações inerentes aos próprios sensores, oclusões e ainda com o elevado número de *DoF* da mão humana.

Esta tese consiste numa solução capaz de fazer o reconhecimento ao vivo de gestos estáticos e dinâmicos, tendo como alvo estações de trabalho de montagem cooperativa. Os dados foram obtidos utilizando as *Senso Gloves* e o *Leap Motion*. O reconhecimento foi feito com base em cinco características distintas da mão humana.

O reconhecimento de gestos estáticos é feito utilizando *Support Vector Machines*. Apesar das metodologias estado-da-arte para estimação de pose utilizarem redes neuronais convolucionais, o sistema desenvolvido é capaz de reconhecer gestos estáticos com valores de precisão superiores a 80%.

Dada a sua natureza temporal, os gestos dinâmicos são modelados por *Modelos Ocultos de Markov*, decisão motivada pelas semelhanças existentes entre a linguagem gestual e o discurso oral. Para treinar mais eficazmente os modelos, o algoritmo *k-Means* é utilizado. Um otimizador de *clusters* baseado no valor *Silhouette* é também utilizado para melhorar o *clustering* dos dados. O sistema desenvolvido é capaz de reconhecer mais de cinco gestos dinâmicos distintos com valores de precisão superiores a 75%.

O desempenho da solução apresentada foi avaliada com recurso a técnicas de validação cruzada, aplicadas sobre um conjunto de gestos criado pessoalmente para operações de montagem colaborativa, permitindo que robôs respondam a comandos e cooperem com humanos em linhas de montagem através de comunicação gestual.

Palavras-Chave

robótica, visão por computador, fusão de dados, gestos estáticos, gestos dinâmicos, reconhecimento de gestos, colaboração humano-robô, *machine learning*

Acknowledgements

To both my supervisors for patiently guiding me.
To my family for supporting me.
To my friends for keeping me motivated and distracted.

Pedro Melo

"Acima da verdade estão os deuses.
A nossa ciência é uma falhada cópia
Da certeza com que eles
Sabem que há o Universo."

Ricardo Reis, *in "Odes"*.

Contents

List of Figures

List of Tables

Abbreviations

3D	3-Dimensional
API	Application Programming Interface
CSV	Comma Separated Values
DoF	Degrees of Freedom
HMM	Hidden Markov Model
HSV	Hue, Saturation, Value
IMU	Inertial Measurement Unit
IR	Infrared
ISO	International Organization for Standardization
LED	Light Emitting Diode
ROS	Robot Operating System
SDK	Software Development Kit
SLS	Structured Light Systems
ToA	Time of Arrival
ToF	Time of Flight
RGB	Red, Green, Blue
SVM	Support Vector Machines
UML	Unified Modelling Language
USB	Universal Serial Bus

Chapter 1

Introduction

The mechanical and cognitive capabilities of robotic systems are growing at an increasing rate, constantly expanding their range of applications. They can be built to perform "tasks that are perceived by humans as dirty, dangerous or dull" [dGA16] and can be found not only inside our homes, cleaning and vacuuming our floors, cooking our meals, but also in the external world, entertaining us, helping the military forces to protect civilian lives or aiding surgeons during medical procedures, just to mention a few examples. With future technology improvements robots will also be likely to also perform tasks foreseen for humans as pleasing, although recent studies [TJN08] show that people prefer to see tasks related to art, evaluation, judgment and diplomacy being performed by other human being.

Our civilization has been automating tasks for a long time, becoming increasingly more proficient since the first and second industrial revolution, which culminated with the development of the first industrial robot in the early 60's by George Devol for factory automation [uni]. Nowadays, industrial robots are highly responsible for the existent mass production of goods with lower costs, since they are faster and more powerful than humans in executing repetitive mechanical tasks. However, the current industrial manufacturing processes must change since "consumers are looking for customized products and services"[WHX17]. Industry 4.0 presents a solution to the industrial problem, by applying better intelligent technologies and by allowing a greater intervention of smart machines such as robots, in the manufacturing process [ZLZ15].

For security reasons industrial robots do not tend to work alongside with human workers in assembly lines. Generally human and robot workplaces are indeed separated in both time and space [FBS15]. This happens because the machines typically don't actively perceive their surrounding environments, merely executing a preprogramed sequence of actions, blindly moving parts at great velocities and manipulating dangerous tools that can easily cause physical harm to a human body or even take a life. In the automobile industry, for instance, human workers are completely excluded from the production lines where robots execute assembly steps, for the reasons mentioned [KSGV+09]. Due to the need to ensure human safety in industrial environments, the ISO standard

10218-1:2011 was created. This document specifies both the requirements and guidelines to follow for the inherent safe design, protective measures and information for use of industrial robots [ISO17].

The so called *Industry 5.0* trend states that allowing humans and robots to cooperate in assembly lines would not only lead to an increse in productivity but also allow the manufacturing of highly customized products[Sit]. Although humans respond and adapt to new tasks quicker, they are slower and have less power than robots, and are not able to produce with constant quality. Despite robots being more powerful and very fast they might not be able to adapt quickly to changes or even adapt at all. So human-robot cooperation in industrial environments can be interpreted as a way to combine qualities, for a greater good. Industrial assembly processes could highly benefit from such cooperation by having the human operator dealing with the variability in the production while taking care of tasks that involve manipulation of sensitive, flexible or hard to assemble components, while having a robot performing operations that require heavy lifting or precise part placement. Another possible use case is the preparation of assembly kits.

Even though cooperative robots already exist at the time of writing, they work mainly in a master-slave scheme, where the machine, the slave, blindly repeats learned actions taught by the human worker, the master, through physical demonstrations. The *LBR iiwa* robot arm developed by *Kuka* [1] is such an example. In order to truly cooperate with humans, robots must be able to communicate with their human co-workers in a way easily understood by the machine but still be the most natural as possible for the human beings. Humans tend to communicate most naturally by articulating gestures using their hands. In fact, when expressing themselves verbally, humans tend to gesticulate without even noticing it. By actively perceiving the world around and recognizing as well as interpreting hand gestures performed by their human co-workers, robots could start cooperating with humans during industrial assembly stages and therefore perform the associated tasks.

Currently there are already several technologies built that provide hand related data, from infrared light based sensors to glove-sensing devices. Examples of such devices are the *Leap Motion*[2] and the *Senso Gloves* [3], respectably. Although the presented glove-based solution does not feel completely natural to the human worker as its use implies the attachment of external devices to the human body, it does not suffer from occlusion problems and has a wider working area in comparison with the presented infrared light based solution that in addition is sensitive to light conditions. It should be noted that either the referred devices are targeted for multimedia applications, hence not being suited for use in real industrial scenarios. However, when used as a proof of concept, the discussed technologies provide for a cheaper and still reliable mean to prototype a gesture recognition system targeting such scenarios.

The hand related data provided by such devices is too raw to be directly used by a gesture recognizer. It is then necessary to first process the hand data using machine learning techniques

[1] https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa
[2] https://www.leapmotion.com/
[3] https://senso.me/

to refine the data in a multitude of useful ways. There are already several open source software libraries dedicated to machine learning and data mining.

Equipped with such technologies robot could be programmed not only to recognize human made gestures, but also being able to cooperate with human workers during a workpiece assembly leading to an overall improvement in productivity in a set of tasks.

1.1 Context

This work was developed in partnership with the C*entre for Robotics in Industry and Intelligent Systems* (*CRIIS* [4]) of *INESC TEC* given their experience in industrial automation and the development of collaborative robotic systems.

1.2 Research Questions

The research questions to be answered with this work are the following:

- **How to recognize static gestures?** Since static gestures don't change in time the samples obtained of the hand data are not correlated. The problem of static gesture recognition can then be seen either as a pose estimation problem or a classification one, where each sample of hand data obtained must be classified according to a properly trained model.

- **How to recognize dynamic gestures?** Unlike static gestures dynamic gestures change in time and therefore distinct samples of hand data can be correlated. At the same time a dynamic gesture is hardly performed the exact same way twice, leading to robustness issues. Instead of individually analysing each obtained sample of hand data, it becomes necessary to cluster the mentioned data into the smaller movements that compose the overall gesture, by analysing the data inner variability. The sequence of obtained clusters must always be similar for a given gesture regardless of how it was performed. In order to allow the gesture distinction, used statistical models must be able to provide a probabilistic insight about what is the most probable gesture being performed by the human worker at a given time.

- **How to obtain intuitive and ergonomic vocabulary of gestures for assembly operations** In order to ease the recognition process, all gestures in the dataset must be as distinguishable as possible, while still making sense to the human operator. The gesture distinction can be optimized by choosing a set of relevant hand features that allow to distinguish every possible hand pose. By mimicking commonly used hand gestures and standardized military gestures, a set of meaningful gestures can be obtained.

[4] http://criis.inesctec.pt/

1.3 Contributions

The goal of this work is to develop a system able to recognize both static and dynamic gestures using the *Senso Gloves*, targeting human-robot collaboration in industrial scenarios. The hand data provided by the *Senso Gloves* is processed and refined using multiple machine learning algorithms which will allow each individual gesture to be recognized by training proper statistical models.

The presented solution is a complement to a framework for human-robot cooperative assembly capable of semantic learning by demonstration that is also being developed at *CRIIS*.

State-of-the-art methodologies were taken as an inspiration for the proposed solution. To the author's knowledge, there is no publicly available dataset for object recognition using both the same set of gestures and the same capturing device, whereby a ground truth was personally created.

Since many state-of-the-art methodologies for gesture recognition use the *Leap Motion* device, a similar dataset using this technology was also created using an equivalent set of hand features, as provided by the device. This allowed, not only to compare the system's performance under distinct capturing devices, but also to compare its performance, at some extent, to already existing solutions.

Problems regarding the way the robot must act upon recognizing a gesture fall out of scope of this project.

The development of the mentioned system aimed to improve human-robot collaboration by allowing live gestural communication between human and machine during assembly stages.

1.4 Outline

This document is structured as following:

- In chapter 2 a summary of the existing and most relevant technologies in use for either hand data acquisition, robot grasping and machine learning is made;
- In chapter 3 a revision and summary of state-of-the-art methodologies for hand tracking and gesture recognition is done;
- In chapter 4 the architecture of the developed system is described;
- In chapter 5 the conducted experiments and the obtained results are exposed and discussed;
- In chapter 6 the final considerations about the work done are made.

Chapter 2

Technologies

This chapter presents relevant technologies used for hand tracking, pose estimation and machine learning. A description of glove-based and *3D* image acquisition systems is also done, as well as an overview of commonly used robotic grippers. Available software libraries are also addressed.

2.1 Camera Systems

A few years ago, cameras that could provide depth measurements, necessary for *3D* imaging, were very expensive. Nowadays there are already available several camera systems that, in spite of being cheaper, allow for depth measurements within good precision and quality levels.

In this section, an overview of existing and relevant *3D* imaging camera systems is made.

2.1.1 Stereo Vision Systems

Stereo vision systems rely upon a pair of cameras to derive depth information about a given scene, mimicking the internal workings of the human visual system [PKA07]. Such systems work on the principle of pixel comparison in both the acquired images. If two pixels match, corresponding to the same object, a point position in the *3D* scene can be computed using the triangulation principle [HRH08]. A graphical representation of a stereo vision system working principle can be seen at figure 2.1. Widely used in the development of intelligent robots, these systems are initially superior in terms of safety, undetectable characteristics, cost, operating range, and reliability, comparatively to other methods [JCP+10]. However, such systems cannot be used for scanning homogeneous surfaces and demand high processing power and time when pixel matching can't be easily done.

2.1.2 Time-of-Flight Systems

Time of Flight (*ToF*), also known as *Time of Arrival* (*ToA*), systems work by casting light into a scene and obtain distance measurements using reflections of the projected light. The projected

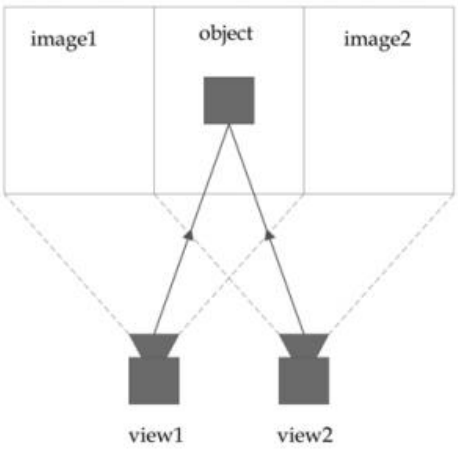

Figure 2.1: Working principle of a stereo vision system. From [HRH08].

light invisible to the human eye, belonging in the *IR* region, is then captured by an imaging sensor operating in the same region. In figure 2.2 the working principle of a *ToF* system is depicted.

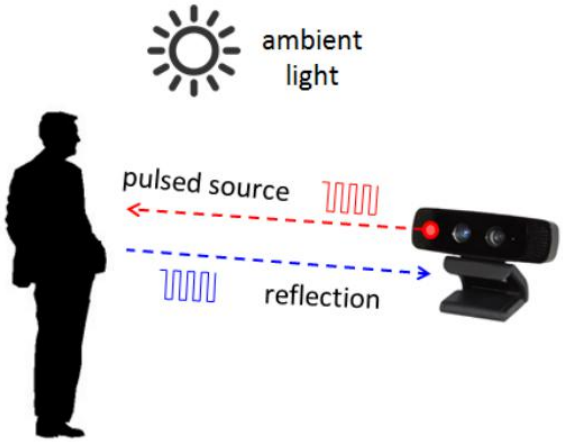

Figure 2.2: 3D *ToF* camera operation. From [Li14].

It should be noted that the projected light is modulated, mainly by square wave modulation, and that reflected light captured by the system has a phase shift. By translating this shift into distances, depth information can be obtained [PKA07].

2.1.3 Structured Light Systems

Systems that use light patterns in order to obtain *3D* information about a given scene are named *Structured Light Systems* (*SLS*), and consist of a projector and a camera [VM98]. The projector casts a coded light pattern, typically made of numbered stripes, upon the scene which is later photographed by the camera. By observing the pattern deformation in the captured scene and mapping pixels in the obtained image to a stripe, spacial information about can be derived [PKA07]. In figure 2.3, a graphical representation of the described process can be seen.

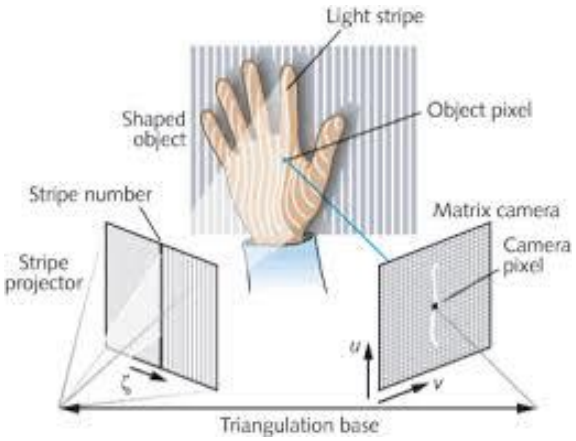

Figure 2.3: Working principle of a structured-light system.

2.1.4 Example Camera Systems

2.1.4.1 *Leap Motion*

The *Leap Motion* is a *USB* device targeting multimedia applications that is able to track and provide hand data without any physical contact.

Figure 2.4: *Leap Motion* device.

The device consists of two cameras and three *IR LEDs*. The device uses the mentioned cameras to track the *IR* light emitted by the *LEDs* (wavelength of 850 nanometres), then using the stereo triangulation for detecting the location of the nearby objects in the scene. This extracted raw data is then passed into a software layer, where it is processed in order to extract the desired hand features. During this process, data fusion techniques are used to "ensure smooth temporal coherence of the data" [Col].

The last version of the controller has an interaction space of eight cubic feet and is able to detect both hands up to a height of 80 centimetres relative to the device.

2.1.4.2 *Kinect*

Microsoft Kinect is widely used in projects that require *3D* imaging ever since it was initially developed for the *XBox 360* ™ video game console. A special adapter to connect the system to a regular computer was even manufactured, allowing the programming of applications where the same could be used, outside the *XBox* development framework. Although its manufacturing is discontinued, *Kinect* technology revolutionized the field and is still supported by many open-source software libraries. The most similar products currently available are produced by *Orbbec*[1].

Kinect V1

Initially the *Kinect* worked as a *SLS* system, containing one infrared projector along with an *RGB* camera, for adding colour to the acquired point clouds, in the same board, being also equipped with a multi-array microphone. The projected pattern, using infrared light rays, can be seen in use in the figure 2.6. The mentioned device can be seen in figure 2.5.

Figure 2.5: *Microsoft Kinect V1*

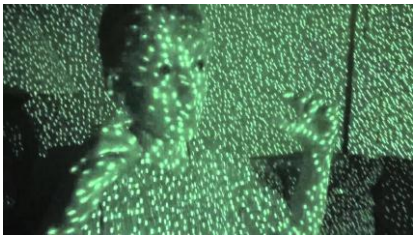

Figure 2.6: *Kinect V1* pattern projected upon a person.

Kinect V2

For its second version, *Kinect* was transformed into a *ToF* system, being more accurate than its predecessor. Besides having an adjustable sensing range, *Kinect* can also capture point cloud data in a wide range of lighting conditions (as long as the projected *IR* pattern is still visible by the *IR* camera). The mentioned device can be seen in figure 2.7.

Figure 2.7: *Microsoft Kinect V2*

[1] https://orbbec3d.com/

2.1.4.3 *Intel RealSense D435* Camera

Intel RealSense D435 is an *USB* stereo based camera system that consists of a pair of depth sensors, a *RGB* sensor and an *IR* projector allowing for both *3D* depth and high-definition colour imaging up to a maximum distance of 10 meters. The depth information is obtained by using the two depth cameras that process the *IR* light casted over the scene by the *IR* projector, being "ideal for makers, educators, hardware prototyping and software development" [Int17]. The mentioned device can be seen in figure 2.8.

Figure 2.8: *Intel RealSense D435* camera.

2.2 Glove-based systems

Glove-based systems are composed of a power supply and an array of sensors for data acquisition and processing, all attached and sewed to a glove, representing "one of the most important efforts aimed at acquiring hand movement data"[DSD08]. At the time of writing there are already several commercial examples of such systems.

2.2.0.1 *Senso Gloves*

Senso Gloves are a wireless glove-based system designed for virtual and augmented reality applications. Each glove can provide precise hand and fingers data with low frequency by means of seven *IMUs* installed in each glove. The mentioned *IMUs* are distributed one per each finger, one for the wrist and one for the hand dorsal. The gloves can also provide haptic feedback effect for every single finger, thanks to a vibration motor placed in each fingertip. Each glove communicates with a *Bluetooth v4* connector, having a maximum range of 20 meters. A battery cycle lasts ten hours and the gloves are made of a fragile material. The last model of the *Senso Gloves* can be seen in figure 2.9.

The *Senso Gloves* manufacturer provides several *SDKs*, allowing the development of applications using the system.

Figure 2.9: *Senso Gloves*, model *DK2*.

2.3 Robotic Grippers

In chapter 5 it is said that the developed system is able to recognize a dynamic gesture called *open*, informing the collaborative robot to open its gripper and let go of its content. Besides that, future work suggestions stated in chapter 6 include the addition of dynamic gestures to individually control each one of the robotic grippers to be used during an assembly. In this section, a brief summary of widely used robotic grippers is done.

Grippers are attached to the end of robotic arms enabling object grasping. Contrarily to the human hand, that has the same shape and workings for all human beings, there exists a wide range of robotic grippers. Nonetheless, some robots can have their gripper changed accordingly to the task they are set to execute. The following list, not exhaustive, presents commonly used robotic grippers:

- **Parallel gripper**. Used for gripping objects with flat sides, they work by oppositely moving two parallel plates alongside a same axis in order to grasp the object, either by its outside edges or by applying pressure on the object inside walls. Additionally, they can be equipped with sensor systems in order to improve their functionality, process reliability and productivity. It should be noted that adaptive 2-finger grippers, which fall inside this category, also have the ability to adapt themselves to the objects shape, making them more robust. An angular gripper can be seen in figure 2.10.

Figure 2.10: *Schunk* parallel gripper.[2]

Figure 2.11: *Robotiq* Adaptive 2 finger gripper.[3]

- **Angular gripper**. Being used for holding larger objects and with odd shapes, they work by oppositely moving two plates angularly, approaching the workpiece sideways. Normally, angular grippers can be adjusted to various angles according to the workpiece to grasp. They can also be improved by equipping external systems, similarly to parallel grippers. An angular gripper can be seen in figure 2.12.

Figure 2.12: *Schunk* angular gripper.[4]

- **Adaptive-3 finger gripper**. Being used for grasping mainly cylindrical shapes, they allow for an even greater flexibility and reliability for any given application, while still allowing for and adaptable grasping. An adaptive-3 finger gripper can be seen in figure 2.13.

Figure 2.13: *Robotiq* Adaptive 3 finger gripper.[5]

[2] https://schunk.com/de_en/gripping-systems/category/gripping-systems/schunk-grippers/parallel-gripper/

[3] http://blog.robotiq.com/topic/2-finger-robot-gripper

[4] https://schunk.com/de_en/gripping-systems/category/gripping-systems/schunk-grippers/angular-gripper/

[5] http://support.robotiq.com/display/IMB/Home

- **Magnetic gripper**. Is used to grasp ferromagnetic materials, working either by having an electromagnet which is turned on and off to accordingly grasp or release the object or by having an enclosed permanent magnet. In the latter case, the release and grasping operations are done by moving the magnet with compressed air inserted sideways. It does not allow for good tracking and pose estimation. An example of magnetic gripper can be seen in figure 2.14.

Figure 2.14: Goudsmit magnetic gripper.[6]

- ***Versaball* gripper**. Consists in a ball, filled with sand like material that softens when air is pumped into it. While at this state, the ball is pushed against the target object, surrounding it and acquiring its shape. The ball is then hardened by removing the air through a vacuum system, allowing it to grasp the desired object. Being generic, this gripper does not allow for good tracking and pose estimation. The mentioned gripper can be seen in figure 2.15.

Figure 2.15: *Versaball* gripper.[7]

2.4 Software Libraries

In order to allow the system training and enable gesture recognition, the obtained hand data must be processed. Nonetheless, a way to communicate the recognized gesture to the cooperative robot must also be established. In this section an overview of popular and relevant software frameworks and libraries for machine learning and robotics is made.

[6] http://www.goudsmitmagnets.com/industrial-magnetic-systems/magnetic-handling/robot-end-of-arm-tooling/magnetic-grippers

[7] http://empirerobotics.com/products

2.4.1 *Robot Operating System*

The *Robot Operating System*[8] (*ROS*) is an open-source set of software libraries widely used in robotics application, aiming to ease robot development. *ROS* logo can be seen in figure 2.16. *ROS* "provides standard operating system facilities such as hardware abstraction, low-level device control, implementation of commonly used functionalities, message passing between processes, and package management"[MF13].

Figure 2.16: *ROS* logo.

ROS was used in the developed system to communicate with the collaborative robot, sending the instructions it must follow after a given gesture was recognized.

2.4.2 *Weka*

The *Weka*[9] is an open-source collection of machine learning algorithms for data mining tasks implemented in the Java programming language, containing tools for data pre-processing, classification, regression, clustering, association rules and visualization. *Weka* logo can be seen in figure 2.17.

Figure 2.17: *Weka* logo.

The *Weka* was used to add robustness to the system, by using its implementation of the *k-Means* algorithm for clustering the obtained hand data during the training and testing of the dynamic gesture recognizer module.

2.4.3 *KValid*

KValid[10] is a free and open-source clustering evaluation package designed for the *Weka* library, containing implementations of the *Silhouette* and *Elbow* methods. *KValid* was used in the application to optimize the number of clusters obtained by the *k-Means* clusterer during the training and testing of the dynamic gesture recognizer module.

[8] http://www.ros.org/
[9] https://www.cs.waikato.ac.nz/ml/weka/
[10] https://github.com/Theldus/KValid

2.4.4 *Java-ML*

Java-ML[11] is an open-source collection of machine learning algorithms for data manipulation, clustering, feature selection and classification. Aiming for simplicity of interface, it was implemented in the Java programming language. This library was used in the developed system as a provider of more versatile data structures for machine learning (comparatively to the ones provided by *Weka*), namely the *KD-Trees* used in the gesture dictionary module.

2.4.5 *LIBSVM*

The *Library for Support Vector Machines*[12] (*LIBSVM*) is a free integrated software for support vector classification, regression and distribution estimation, that also supports multi-class classification. Currently, there are available several implementations of this library in several programming languages. Its implementation in the *Java* programming language was used for training and testing the static gesture recognizer module.

2.4.6 *Jahmm*

Jahmm [13] is an open-source collection of algorithms regarding the training and querying of *Hidden Markov Models* (*HMM*), implemented in the *Java* programming language. It also provides some machine learning algorithm. *Jahmm* was the core element of the implemented module for dynamic gesture recognition.

A theoretical discussion regarding *Hidden Markov Models* is made in the following chapter.

[11] http://java-ml.sourceforge.net/
[12] https://www.csie.ntu.edu.tw/~cjlin/libsvm/
[13] https://github.com/KommuSoft/jahmm

Chapter 3

Related Work

In this chapter state-of-the-art methodologies for hand tracking, pose estimation and gesture recognition are discussed.

3.1 Hand Tracking and Pose Estimation

In this section the state-of-the-art approaches for tracking and pose estimating of both human hands and robot grippers are discussed.

3.1.1 Human Hand Modelling

Many state-of-the-art approaches for hand tracking and pose estimation *require the use of artificial 3D hand models* that are overlapped upon the observed data in order to evaluate a solution's quality. This evaluation is carried out by a cost function, whose performance depends highly on the type of hand model used.

The first step towards human hand modelling is the acquisition of its kinematic structure, revealing which are the articulated hand parts, the existing connections between themselves and how they move. All possible hand movements occur around the finger joints and the wrist, leading to a hand kinematic structure as depicted in the figure 3.1, where each dot corresponds to a different joint and the bottom square to the wrist. The human hand accounts for a total of 27 *DoF* [NBT07]. Nonetheless some authors use variants of the presented structure having small differences in total *DoF*, in order to reduce the problem dimensionality and increase performance.

In "order to model the hand, there is a variety of options that depend on the balance required between accuracy and performance" [Bar16]. These options are:

- **Sphere based hand models**. Built uniquely from spheres. Despite being simple, are not anatomical correct.

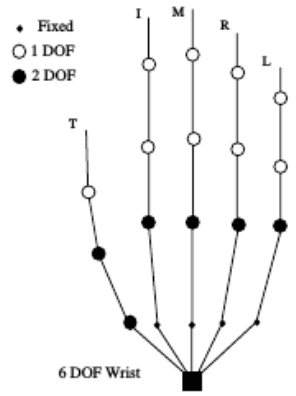

Figure 3.1: *Hand kinematic structure. From [NBT07].*

- **Geometry based hand models**. Built upon basic geometric primitives. An example of such model can be seen in figure 3.2.

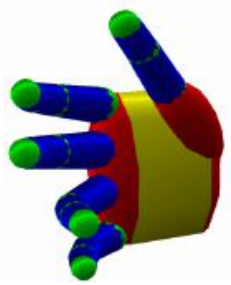

Figure 3.2: Example of geometric based hand model. From [IOA11].

- **Mesh based hand models**. Built exclusively out of meshes. Being the most complex type, they have greater anatomically accuracy. Also require higher computer power. Figure 3.3 show an example of such hand model.

Figure 3.3: Example of mesh based hand model holding an object. Extracted from [SMZ+16].

Additionally, constraints regarding finger extension and movements are usually applied to the chosen model, in order to reduce the state space of possible hand positions, simplifying both the tracking and pose estimation. The use of such restrictions also proves useful when dealing with occlusions, as explained in the following sections.

3.1.2 Robot Gripper Modeling

As previously stated, robot grippers exist in a variety of shapes and work in several distinct ways. This leads to complicated issues regarding robot grippers modelling. As a matter of fact, in [SPM10] it is claimed that "little attention has been put on the use of vision when the robot hand makes the first contacts on the objects".

3.1.3 Pose Estimation Pipelines

"Many algorithms used in hand tracking have their roots in methods proposed previously in human body tracking" [NBT07]. Nowadays, every approach to the problem fall in one of three possible classifications [Bar16]:

- **Appearance based approaches**. Also known as *discriminative*, they estimate "articulated motion states *directly from images* by learning the mapping from an image feature space to the object state space" [CCH05]. Being less computationally expensive, they require large amount of training data in order to perform full hand pose estimation, being very used in gesture recognition. The typical pipeline of such approaches can be seen in the figure 3.4.

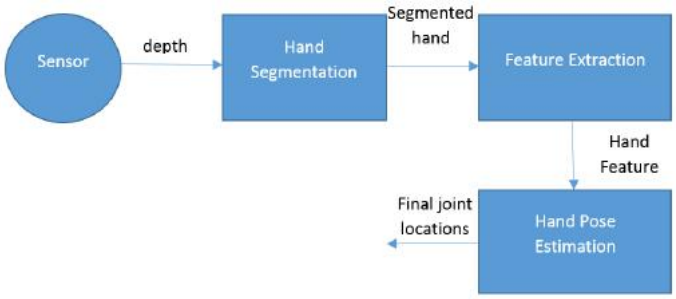

Figure 3.4: Pipeline for an appearance based approach. From [Bar16].

- **Model based approaches**. Also known as *generative*, they try to optimally fit hypothesized hand models with the observed data using a cost function. Thus, previously generated hyphotesis are taken into account during the calculation of new plausible hand poses. The resulting hand pose corresponds to the hypothetic hand pose that is closer to the observed one. An example generative pipeline can be seen in the figure 3.5. As stated in the previous section, performance can be tuned by either changing the hand model or reducing the number of *DoF*.

- **Hybrid methods**. Combine both the previous methodologies. Initially an appearance-based pipeline is used to obtain a hand pose which is then passed to a model-based pipeline.

3.1.4 Hand Acquisition

The first step towards hand pose estimation is to process the sensor data in order to distinguish the hand or gripper from the surrounding environment. This is done through image segmentation.

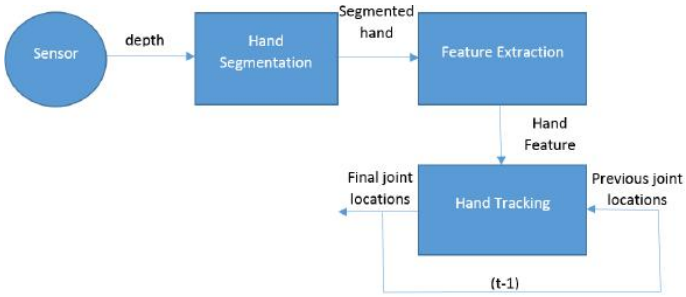

Figure 3.5: Pipeline for a model based approach. From [Bar16].

3.1.4.1 Colour Segmentation

The human skin has a very distinct colour allowing for hand detection via colour segmentation. However, images obtained by cameras tend to use the *RGB* colour space which are very sensitive to illumination changes. Authors also use alternative colour spaces, namely the *HSV* [SMZ+16] and *YCbCr* [dCM06] spaces, since they both separate the colour and light intensity components, being more robust to illumination changes. When nearby objects or background surfaces have colours similar to the skin colour, this type of segmentation becomes unpractical.

Some authors resorted to using either hand paint or simple colourful gloves [WP09]. In spite of easing the colour segmentation process, these approaches are considered intrusive. However, if the worker is obligated to use gloves in the workplace, such solutions should be taken into account.

3.1.4.2 Depth Segmentation

Since depth imaging systems became widely available, researchers started developing ways to use depth data to recognize and track hands. Comparatively to colour segmented approaches, the hand can be away from the camera in dark environments and still be detected, leading to illumination and distance invariant hand detectors. This is depicted in the figure 3.6.

Figure 3.6: Comparison between colour and depth images in a low illuminated scenario. Adapted from [PYK+12].

Considering the hand as closest to the camera, a simple segmentation can be done based on depth values, as done in [MN06]. However, due to light reflections, the edges of objects usually get noisy, complicating the hand recognition process. This problem can be surpassed using background subtraction techniques followed by morphological operators and mean-shift[WMC+15],

in order to obtain only the moving parts. It should be noted that such solutions assume that only the hand to track is moving.

3.1.4.3 Machine Learning Techniques

By using point clouds provided by a *Kinect* camera, a novel way for *3D* scene segmenting was presented in [CPCC16], using linear regression algorithms, that allow the classification of cloud points as belonging or not to the hand. It should be noted that present techniques are also applicable to *3D* object segmentation and recognition.

3.1.5 Tracking and Pose Estimation

Generative approaches tend to use either variants of the *PSO*[IOA11] and the *ICP*[CPCC16] algorithms, since they transform the pose estimation problem into an optimization one. The *PSO* algorithm works by having a swarm of particles randomly disposed in a state space, each one searching for local function minimum. The best minimum found up to a given moment is known by all the particles that move in the direction of that point. Once a better minimum is found, the mentioned value is update. The algorithm ends when all the particles arrive at the same point. *ICP* algorithm is similar to the *PSO* algorithms, minimizing the error in a distinct way. In spite of being computationally expensive, these approaches were the first to allow continuous hand tracking and pose estimation. As previously noted, the chosen 3D hand model type influenced the results.

A few years ago, discriminative approaches only allowed for pose estimation of finite sets of related hand positions, being widely used in gesture recognition. This was due to the reduced amount of training data available. 3D point cloud data was then processed using both clustering and data fusion algorithms [WMC+15][PYK+12]. Nowadays, due to recent developments of machine learning and the existence of relatively large image datasets, both state-of-art discriminative and hybrid approaches are based upon deep learning technologies. In [DYZ+17], [WPVY17] and [DWÖG17] a *CNN* is trained over existing datasets in order to obtain pose estimation from depth information, with success. Major disadvantage of such solutions include the high volume of needed training data.

3.1.6 Occlusion Handling

The tracked hand can suffer from the following types of occlusion:

- **Partial Occlusions**. Occur when fingers overlap or cross, occluding either the palm or each other;
- **Global Occlusions**. Occur when portions of the hand surface are occluded by objects being held, as depicted in figure 3.7.

Both occlusions can be surpassed by applying probabilistic models over the hand anatomy and kinematic structure. Such is the case in [HSKMG09], where *Markov Fields* are used to estimate

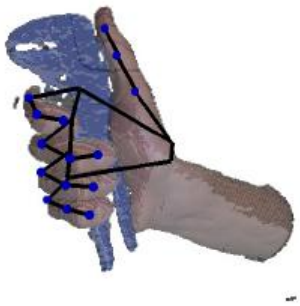

Figure 3.7: Global hand occlusions. From [HSKMG09].

the pose of occluded hand joints. This approach is feasible since a joint's pose only depends on its adjacent joints' poses. It then suffices to only detect a subset of joints for each finger.

3.1.7 Datasets

The field of hand tracking and pose estimation "lacks a systematic public benchmark for fair evaluation of different methodologies" [YYGK17], since existing datasets are either too small or not compliant with each other. To overcome the related dataset dimension problems, data augmentation mechanisms were proposed in [DYZ+17].

The table 3.1 contains a summary of existing datasets at time of writing. Particular information and citations about each dataset can be found at [YYS+17].

Dataset	Frames	Joints	Subjects	View point	Depth map resolution
Dexter 1	2,137	5	1	3rd	320x240
MSRA14	2400	21	6	3rd	320x240
ICVL	17604	16	10	3rd	320x240
NYU	81009	36	2	3rd	640x480
MSRA15	76375	21	9	3rd	320x240
UCI-EGO	400	26	2	ego	320x240
Graz16	2166	21	6	ego	320x240
ASTAR	870	20	30	3rd	320x240
HandNet	212928	6	10	3rd	320x240
MSRC	102000	22	1	3rd	512x424
BigHand2.2M	2200000	21	10	full	640x480

Table 3.1: Summary of existent field datasets at time of writing. Adapted from [YYS+17].

3.2 Gesture Recognition

In this section state-of-art techniques used for gesture recognition are discussed.

3.2.1 Gesture Classification

Gesture-based computer interaction can nowadays be done in a multitude of ways that differ in diversity, complexity and spontaneity [VMSLB13], thus leading to the creation of several possible gesture taxonomies, based on different classification criteria. In [Kar05], the following type-based gesture classification is presented:

1. **Deictic**. Involve pointing to a physical object or to a space region;
2. **Manipulative**. Involve mimicking with empty hands the movements to be performed by the machine while manipulating a real physical object;
3. **Semaphoric**. Involve performing movements using hands, flags or lights that are catalogued in a gesture dictionary, such that each dictionary entry associates a specific gesture with a specific meaning or action to be carried out after that gesture is performed. Usually require the pre-recording of gestures and a system training phase. Semaphoric gestures can be subdivided into the following way:
   (a) *Static Gestures*. Consist in static poses made by the fingers;
   (b) *Dynamic Gestures*. Gestures whose information can be perceived through movements;
   (c) *Strokes*. Similar to dynamic gestures but consisting of fast stroke-like gesture.
4. **Gesticulation**. Involve gestures that accompany human speech and clarify it. Their meaning is also dependant on the speech's topic and context;
5. **Language**. Gestures used for sign language;

The presented classification also considers the case where a combination of many gestures belonging to multiple of the previously stated classes is done.

By analysing figure 3.8 it is possible to observe the predominance of semaphoric gestures relative to the other individual gesture-type classes.

3.2.2 Hand Data Features

The set of selected hand features must capture the obtained data variability in order to distinguish all the performed gestures.

The first step towards hand position differentiation consists in finding a set of hand features that allow to tell which fingers are stretched, to tell which fingers are "open" and "closed", and how much a finger is stretched. The set of possible hand features used in the developed system for the stated purpose was the following:

- The distance between each fingertip and the hand palm (F1);
- The angle between two neighbour fingers (F2).

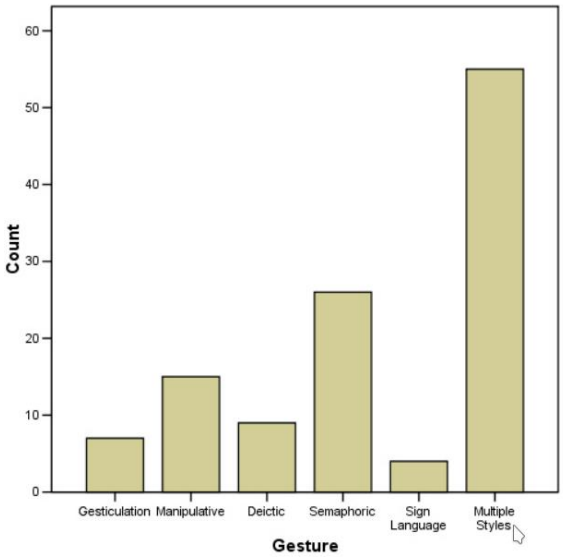

Figure 3.8: Distribution of the different gesture styles. Extracted from [Kar05].

It is possible for a performed gesture to have different meanings according to the orientation of the hand palm and the wrist. Therefore doing the same gesture with the palm facing upwards or downwards results in two distinct gestures. The set of hand features used for the stated purpose was the following:

- Normalized coordinates of the normal vector to the hand palm (F3);
- Normalized coordinates of the normal vector to the operator's wrist (F4).

The direction in which the hand is moving must also be taken into account, especially when dealing with dynamic gestures. For stated purposed the following hand features were used:

- Normalized coordinates of the hand palm velocity vector (F5);

Vector coordinates can take any value within a given range, leading to robustness issues. However, the information they carry is too precious to the recognition process to just be discarded. If only the magnitude of the vector coordinates is considered, the vector coordinates can be normalized so that all positive and negative coordinates take a same unique value per class. This preserves not only the information regarding the palm and wrist orientation, but also the hand direction of movement while not compromising the system robustness.

3.2.3 Algorithms for Gesture Recognition

The developed system only detects semaphoric gestures. The algorithms used for recognition of static and dynamic gestures (both *pure* dynamic and strokes) differ from each other, since the two types of gestures differ in nature. In this section state-of-the-art gesture recognition algorithms for each gesture type will be explained.

3.2.3.1 Static Gesture Recognition

Static gestures don't change in time and thus the samples of obtained hand data are not correlated. This characteristics allow us to treat the static gesture recognition problem as another hand pose estimation problem which can be solved using the previously mentioned hand pose estimation methodologies, namely through the use of neural networks.

Static gesture recognition can also be seen as a classification problem, where each sample of obtained hand data must be labelled according to a trained model.

Due to the fact that high amounts of data are required for proper neural network training and that the gesture dataset used is personal and has low volume, a suitable classification algorithm was chosen for static gesture recognition. Due to its simplicity, the *Support Vector Machines* (*SVM*) techniques were chosen.

3.2.3.2 *Support-Vector Machines*

Support-Vector Machines, also called *Support Vector Networks*, is a supervised technique for classification and regression that predicts whether data falls into one of possible categories, according to the training data. *SVM* does this by mapping "the input vectors into some high dimensional feature space Z through some non-linear mapping chosen *a priori*" [CV95]. Then a hyperplane is used to accomplish the algorithm goal, separating the data with the largest distance to the nearest training data points, called the *optimal margin* of any class thus minimizing the generalization error of the classifier [Wik17]. Figure 3.9 depicts a classification problem in 2D solved using *SVM*.

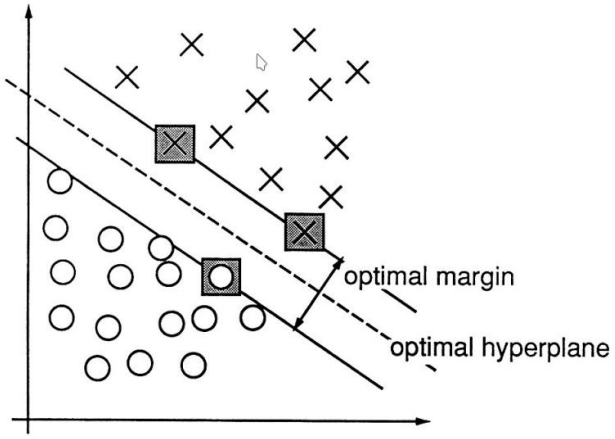

Figure 3.9: A separable problem, with respective optimal margin and hyperplane. Extracted and adapted from [CV95].

Besides linear regression, actual implementation of the algorithm allow for non-linear regression using several available kernels.

3.2.3.3 Dynamic Gesture Recognition

Dynamic gestures change in time and thus distinct samples of hand data can still be correlated in time. Besides that, the same dynamic gesture is hardly performed the exact same way twice, leading to robustness issues. Instead of analysing each obtained sample of hand data, it becomes necessary to cluster the data in a way that maximizes the uniformity of each cluster, in order to remove recording noise and to ease the gesture recognition. Figure 3.10 depicts the results obtained from a comparison between several popular clusterer algorithms under a same dataset of dynamic gestures. Although the *fuzzy c-means* algorithm performs better than the remaining algorithms, the *k-Means* algorithms is suggested, "if a compromise between time and recognition rate is tolerable" [PN09].

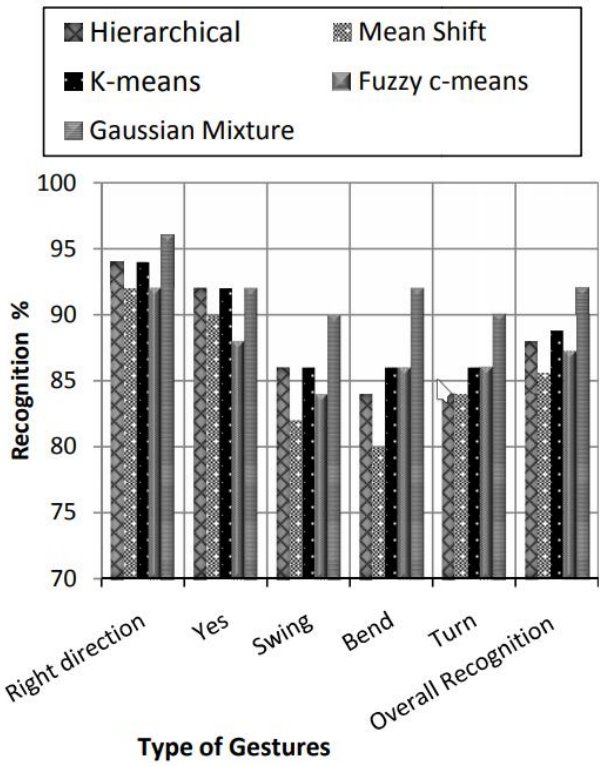

Figure 3.10: Performance of several clusterer algorithms for dynamic gesture recognition. Extracted from [PN09].

In order to allow for gesture distinction, proper statistical models must be able to provide probabilistic insight about what is the most probable gesture being performed by the human worker should be used. Recent researches started relating speech variations to visual gestures, then justifying the use of *Hidden Markov Models*, traditionally used in speech recognition, in dynamic gesture recognition [GVD+16].

3.2.3.4 *k-Means* algorithm

A suitable clustering algorithm for dynamic gesture recognition is the *KMeans* algorithm. *k-Means* is an unsupervised learning algorithm for clustering high-dimensional data set into a number of k clusters which is set *a priori*. It works by following the presented steps [TT11]:

1. Initialize k centroids into the high-dimensional space. The way the cluster centroids are initialized is relevant, so that they should be places as far away as possible from each other;
2. Assign each data point to the cluster that has the closest centroid;
3. Recalculate the k cluster centroids;
4. Repeat steps 2 and 3 until the centroids stop moving.

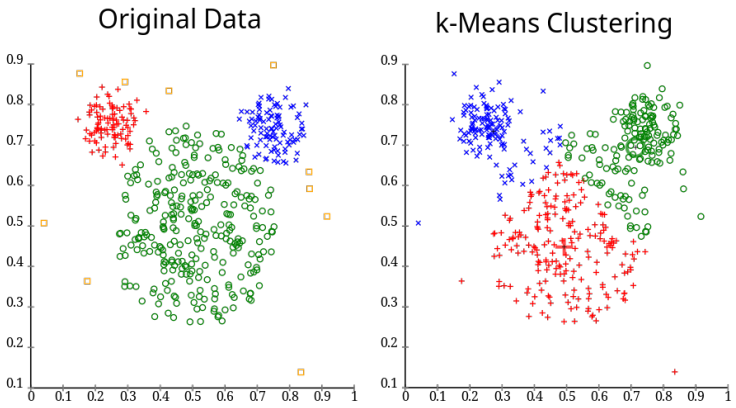

Figure 3.11: Sample data clustering using the *k-Means* algorithm.

Figure 3.11 depicts both the result of applying this algorithm into a sample data set and the optimal clusterization for the given dataset.

An inherent problem with the *k-Means* algorithm is that, in spite of always clustering the data in k clusters, a non-optimal data clusterization can be achieved since the number of clusters k is manually inserted and can be poorly chosen, as seen in figure 3.12.

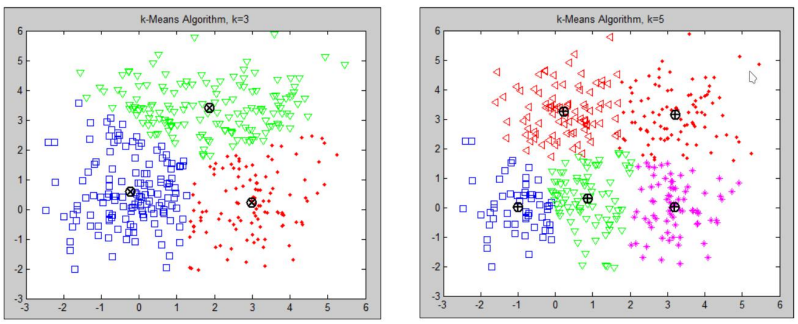

Figure 3.12: A same dataset clustered with *k-Means* using distinct k values. Extracted from [TKD+15].

However, the number of clusters can be optimized using the *Silhouette Measure* [Rou87]. A *silhouette* is a tool used to assess the validity of the clusterization and is constructed to select the optimal number of clusters [TKD+15]. It can be calculated using equation 3.1, where *b(i)* stands for the average distance between the points in cluster *i* and the centroid of the next closest cluster and *a(i)* stands for the average distance between the points in cluster *i* and the centroid of the same cluster. *s(i)* stands for the *silhouette* value while using *i* clusters.

$$s(i) = \frac{b(i) - a(i)}{max\{a(i); b(i)\}} \tag{3.1}$$

By clustering the same data using different values for k and plotting the obtained *silhouette*, the optimal number of clusters *i* can be found where *s(i)* is maximized. Disadvantages of using this method reside in the need to cluster the same data multiple times, which can prove time expensive, especially when using high volume datasets.

3.2.3.5 *Hidden Markov Model*

The statistical model used for dynamic gesture recognition was the *Hidden Markov Model*, due to the fact that gestures themselves can be seen as a sequence of static and distinct hand poses, where the probability of having a given pose of a sequence only depends on the previous pose in the sequence, for a given gesture.

In order to understand *Hidden Markov Models* it is necessary to first understand *Markov Chains*. A Markov chain can be seen as a weighted automaton in which arc's weight represent the probability of travelling from a state to another [JM00], as depicted in figure 3.13.

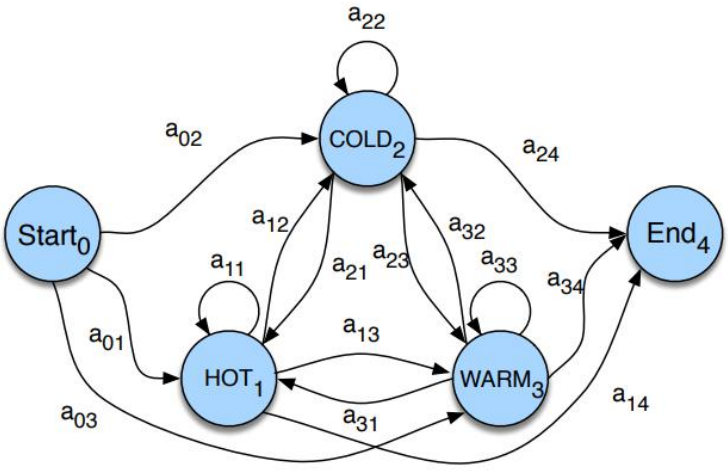

Figure 3.13: Sample *Markov chain* for weather prediction. Extracted from [JM00] .

A *Markov chain* is constituted by the following elements:

- The N states in the model;

- The model state transition probability distribution *A*, which tells the probability of reaching a model state from every other state. It should be noted that for any given state, the sum of transition probabilities must add up to 1;
- The model initial state distribution (π);
- An initial state *q0* and a final state *qf*, independent of any observation.

Additionally, every *Markov chain* assumes the called **Markov assumption**, which states that the probability of being at a given state depends exclusively on the previous state.

Hidden Markov Models are extensions to *Markov chains*, "whose job is to assign a label or class to each unit in a sequence, thus mapping a sequence of observations to a sequence of labels"[JM00]. Besides the previous stated elements, *HMM*s are also made of the following listed elements [Rab89, JM00]. A diagram is presented in figure 3.14:

- The possible and distinct observable symbols *o* from the vocabulary *V*;
- The observation symbol probability distribution *Bj* for every possible state *j*, also called the *emission probabilities*, indicating the probability of being in that state when a given symbol was observed;

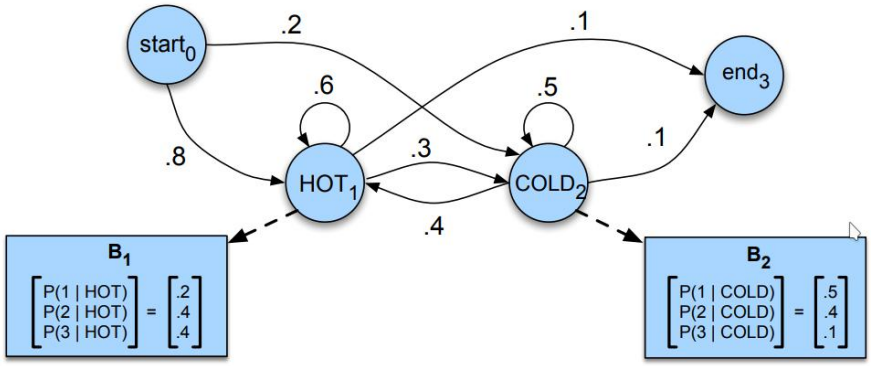

Figure 3.14: Sample *HMM* for weather prediction. Extracted from [JM00] .

It should be noted that the states of a *HMM* are *hidden* and a sequence of states transitions can only be predicted by means of a sequence of exterior and possible observations. Each hidden state can only produce a single observation. Besides the *Markov assumption*, every *HMM* also assumes the called **Output independence assumption**, which states that the probability of an output observation *oi* depends only on the state that produced the observation *1i* and not on any other states or any other observations.

The *Hidden Markov Model* "approach to gesture recognition is motivated by the successful application of hidden *Markov* modelling techniques to speech recognition problems. The similarities between speech and gesture suggest that techniques effective for one problem may be effective for the other as well" [MYS+97].

However, in order to be used in real applications, one must be able to solve the so called *three classic problems* [Rab89] which are:

1. **"How to tell the likelihood of an observation sequence, according to a given model?"** This problem is solved by means of the *Forward* algorithm, which uses dynamic programming techniques to efficiently compute the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence by implicitly folding each of these paths into a single forward trellis [JM00].

2. **"How to tell the most probable sequence of states for a given sequence of observations, according to a given model?"** This problem is solved by means of the *Viterbi* algorithm, which uses dynamic programming techniques to select the state that has the highest probability and backtrack to the path that produced the highest probability using the backpointer and return the states [Kic16].

3. **"How to train a model in order to fit the training data?"** This problem is solved by means of the *Baum-Welch* algorithm, also called the *Forward-Backwards* algorithm that calculates, for each training sequence, the contribution of a sequence to both the transitions of the model and the emission probabilities. New model parameters are updated until a log likelihood is smaller than a given threshold or when a defined maximum number of iterations is passed.

In the developed application only the *Baum-Welch* and *Forward* algorithms are used.

Chapter 4

System Architecture

In this chapter the architecture of the developed system is discussed.

4.1 System Overview

The developed system works by separately detecting static gestures performed by the user's left hand and dynamic gestures performed by the user's right hand. The system starts by analysing data coming from the user's left hand, verifying if the *start* static gesture is being performed. In that case, the system also starts analysing information from the user's right hand in order to recognize any of the trained dynamic gestures. Figure 4.1 depicts the state machine for the system, where:

- **State 0**. The system verifies if the user is performing the *start* static gesture. If so, a *START* event occurs. Otherwise a *STOP* event occurs;
- **State 1**. The system also starts collecting the user's right hand until the user stops performing the *start* static gesture, leading into a *STOP* event;
- **State 2**. The system processes the collected right hand data in order to recognize the most probable dynamic gesture. To do so, the difference between the certainty levels of the two most probable gestures is used, in order to avoid ambiguous results. If the stated difference is greater than 10%, the gesture with greater certainty level is accepted as the most probable and a *DETECTED* event occurs. Otherwise a *GARBAGE* event occurs;
- **State 3**. The system has recognized a trained dynamic gesture, and communicates the robot the proper command to be executed.

For simplicity reasons, the developed system is not able to recognize two dynamic gestures in a row.

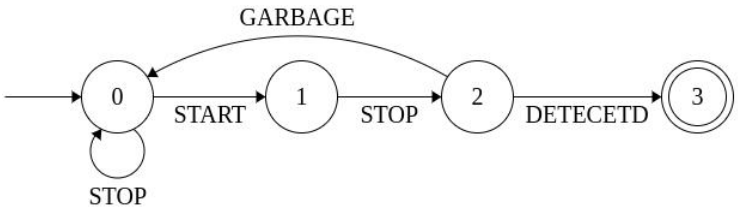

Figure 4.1: System state machine.

4.2 Gesture Dataset

In this section, the used gesture dataset is introduced and data acquisition mechanisms are discussed. As mentioned before, static gestures must be performed exclusively with the left hand, while dynamic gestures must be performed exclusively with the right hand.

Chosen *Senso Gloves* hand features

At the time of writing, the *Senso Gloves* are only capable of directly providing the coordinates at the centre of the hand palm, not giving any direct information regarding the fingers' joints and tips coordinates. This issue complicates the retrieval of features F1 and F2 (see subsection 3.2.2). However, the *Senso Gloves* provide the *Euler angles* for every finger (yaw and pitch only, since it is anatomically impossible for a finger to roll around its central axis). Since yaw is directly proportional to ideal feature F1 and pitch is directly proportional to ideal feature F2, the values of those angles were used as a substitute.

Despite providing relevant hand data regarding ideal features F3 and F4, the *Senso Gloves* do so in quaternion format and not in the desired unit vector format. This issue was solved by transforming the quaternion representation into an axis-angle one. The normal vectors to the hand palm and wrist were then obtained.

The *Senso Gloves* already provides the hand speed in the desired vector format.

Chosen *Leap Motion* hand features

Using the *Leap Motion API* it is already possible to directly obtain the necessary hand data for all features except F2. Nonetheless, the *API* provides built-in vector operations, easing the calculation of the necessary angles. The angles were calculated using the dot product between the vectors that go from the hand palm to the fingers tips.

4.2.1 Static Gesture Dataset

The developed system is trained to recognize only two static gestures. The first, called the *start* gesture, is used to signal the system to start recognizing dynamic gestures. This gesture is done by closing the hand with the palm facing the ground and then by approximating it to the chest, with the hand palm now facing forwards.

Figure 4.2: *Start* gesture

Any other gesture performed by the left hand is perceived as a *stop* gesture, which signals the system to stop processing any right hand data until the *start* gesture is performed again. For training purposes, a total of 100 samples were obtained for each mentioned static gesture. For testing purposes, another set of distinct 100 samples were obtained.

4.2.2 Dynamic Gesture Dataset

The dynamic gestures used are summarized in the following list. It should be noted that every gesture starts with the same closed hand position with the palm facing the ground. For testing purposes, a total of 10 recordings were obtained for each gesture.

- ***Home***. The robot must return to its initial position. The operator must initially approximate the closed hand to his chest, then rotate the hand and open the fingers so that the palm faces the operator's chest. The final step consists in performing an elbow centered counterclockwise rotation of 90 degrees (see figure 4.3);

Figure 4.3: *Home* gesture

- ***Hover***. The robot must go to the position before initializing the placement operations. The operator's hand must be opened and rotated 90 degrees counterclockwise, return to the opened position and then rotate another 90 degrees clockwise, always with the palm turned downwards (see figure 4.4);

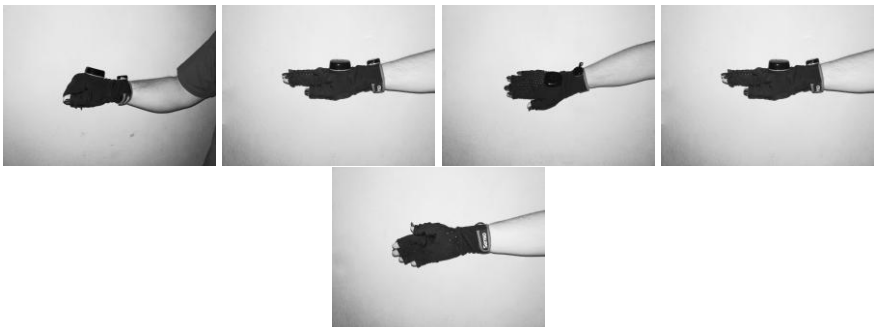

Figure 4.4: *Hover* gesture

- ***Next***. Pass to the next set of instructions. The operator must stretch the index and middle fingers and perform a circular motion clockwise (see figure 4.5);

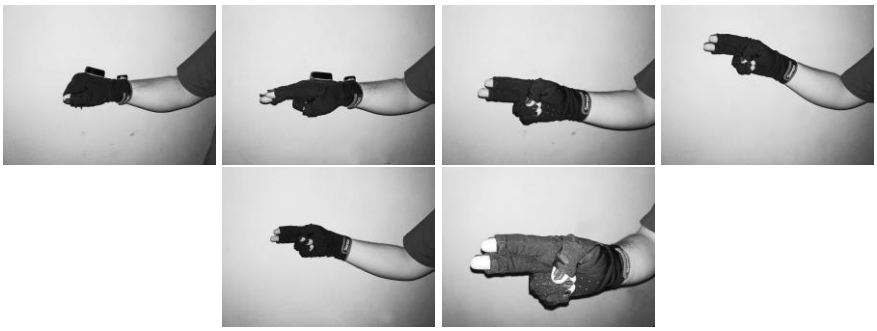

Figure 4.5: *Next* gesture

- ***Open***. Open the robotic gripper. The operator must simply open his right hand, stretching every finger (see figure 4.6);

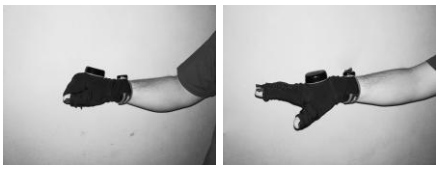

Figure 4.6: *Open* gesture

- ***Previous***. Go back to the previous set of instructions. The operator must stretch the index finger and perform a counterclockwise circular motion (see figure 4.7);

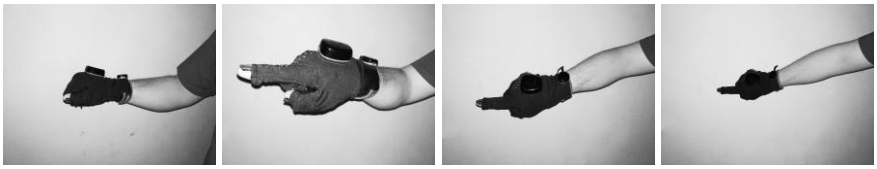

Figure 4.7: *Previous* gesture

- ***Stop***. Stops current operation. The operator must raise the index finger and move it 45 degrees to the left, return to original position and then moving the finger 45 degrees to the right (see figure 4.8).

Figure 4.8: *Stop* gesture

4.3 System Training

In this section the system training sequence is discussed.

Training the Static Gesture Recognizer

The training phase for the static gesture recognizer is schematized in the figure 4.9 and involves the following steps:

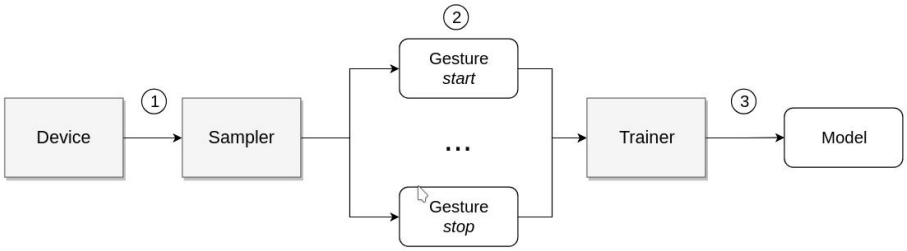

Figure 4.9: Flow of static gesture data during the training phase.

1. The selected device sends continuous left hand data into the sampler module;
2. The sampler module assigns a distinct label to the received hand data, so that each distinct static gesture has its unique label. The labelled data is then passed to the training module;
3. The training module applies the discussed *SVM* techniques upon the received labelled data and outputs a trained model. It should be noted that the RBF kernel provided by *LIBSVM* was used.

Training the Dynamic Gesture Recognizer

The training phase for the dynamic gesture recognizer is schematized in the figure 4.10 and involves the following steps:

1. The selected device sends a continuous stream of right hand data into the recorder module;

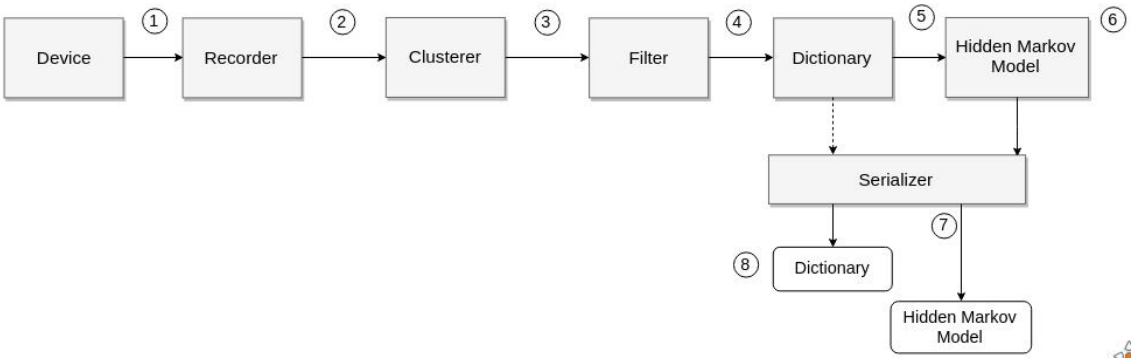

Figure 4.10: Flow of dynamic gesture data during the training phase.

2. The recorder module aggregates the data into separate recordings, so that each recording contains a single and complete performance of the gesture being recorded. All the recordings are then passed into the clusterer module;

3. The clusterer module applies the *k-Means* algorithm using the *Silhouette* method to each of the received recordings in order to optimally cluster the continuous data. Each cluster is tagged with a local *ID*, which ranges from 0 up to the optimal value of *k - 1* and each recording is then converted into a sequence of clusters *IDs*. The obtained local sequence is passed to the filter module;

4. The filter module applies both a filtering step and a compression step to the sequence of local *IDs*. The first step aims to remove existing noise, before it could reach the following modules. Noise is represented by sporadic *IDs* in the sequence and is removed by means of a histogram based filter that applies a fixed sized sliding window over the sequence. For each iteration, the pivot element assumes the value of the most common *ID* in the window. A sliding window size of 9 was used during practical experiments. The most common *ID* is calculated using a frequency histogram which maps each *ID* to the number of occurrences inside the window. The second step aims to improve the training phase by reducing the sequence size. This is accomplished by removing consecutive elements with the same *ID*. Since two non-related clusters can have the same local *ID*, is it necessary to translate local *IDs* into global ones to avoid confusion during the model training. The processed sequence is then passed to the dictionary module;

5. The dictionary module converts the sequence of local *IDs* into a sequence of global cluster *IDs*. For that, it assigns an incremental *ID* to each new distinct cluster centroid. For every cluster in the sequence, the dictionary calculates the euclidean distance to the most similar cluster. If the calculated distance is greater than a specified threshold, then a new entry is added to the dictionary. If the euclidean distance between two cluster centroids is smaller than a specified threshold, the dictionary returns the *ID* of the nearest cluster centroid. To optimize the search for the nearest cluster centroid a *KDTree* data structure was used. The sequence of global cluster *IDs* is then used to train the gesture *HMM*.

6. The *HMM* module uses the sequence of global cluster *IDs* to train the gesture *HMM* using *Baum-Welch* algorithm;

7. After all recording sequences from a given gesture are used to train the *HMM* of that gesture, it is passed into a serializer module. This step allows the models to be trained once and be used later on;

8. After every gesture is serialized, the completely updated dictionary can also be serialized in order to be used during the recognition phase.

4.4 Gesture Recognition

In this section the sequences for gesture recognition are discussed.

Recognizing Static Gestures

The recognition phase for static gestures is schematized in the figure 4.11 and involves the following steps:

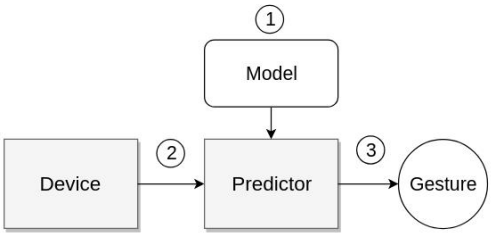

Figure 4.11: Flow of static gesture data during the recognition phase.

1. The model obtained during the training phase is passed to the predictor module;

2. The selected device passes continuous left hand data into the predictor module;

3. The predictor module, based on the passed model, makes a prediction and labels the hand pose. The labels match the same labels used during the training phase.

Recognizing Dynamic Gestures

The recognition phase for dynamic gestures is schematized in the figure 4.12 and involves the following steps:

1. The previously serialized dictionary and *HMM* during the training stage are now deserialized;

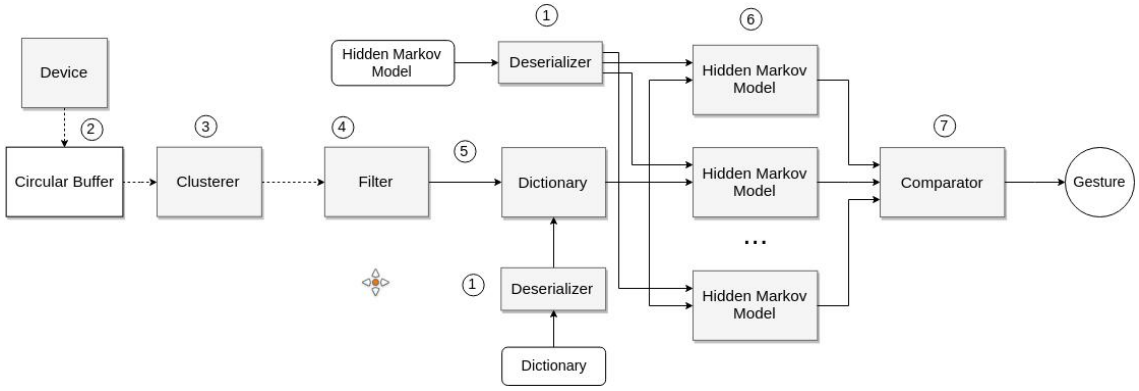

Figure 4.12: Flow of dynamic gesture data during the recognition phase.

2. The hand features are obtained from the selected device and inserted into a circular buffer, creating the necessary temporal sliding window for the recognition of dynamic gestures while limiting the amount of sensor data that is passed. The content of the circular buffer is passed to the clusterer module.

3. The data passed to the clusterer module is clustered and converted into a sequence of local cluster IDs, according to the same processes explained in the previous section;

4. The data is then passed to the filter module to be removed from noise and to be compressed, according to the same processes explained in the previous section;

5. The sequence of local IDs is passed into the gesture dictionary to be converted into a sequence of global cluster IDs;

6. After translating local cluster IDs into global cluster IDs, the sequence of cluster IDs is passed to each of the deserialized *HMM*s;

7. Each gesture model sends the probability for the given sequence of global IDs to the comparator module, calculated with the *Forward* algorithm. The task of the comparator module is to select the most probable gesture based on the passed probability values. Since every sequence of global IDs is evaluated separately by all the gesture models, it is possible for a given sequence to be considered highly probable by more than a single *HMM*, thus leading to an ambiguous conclusion. The comparator module only recognizes a gesture when the difference between the two greatest certainty levels is superior to 10%.

4.5 Packages and Class Diagrams

The developed system was divided in the following manner:

1. ***Clusterers* package**. This package contains the following classes, that constitute the clusterer module:

- ***KMeansCluster* class**. Wrapper for *SimpleKMeans* class and the cluster optimizer provided by the *Weka* and *KValid* libraries, respectably. Interfaces with the *GestureDictionary* class, to which it passes the local cluster IDs;
- ***Translator* class**. Helper class that translates the sequence of cluster IDs using the global IDs provided by the dictionary module.

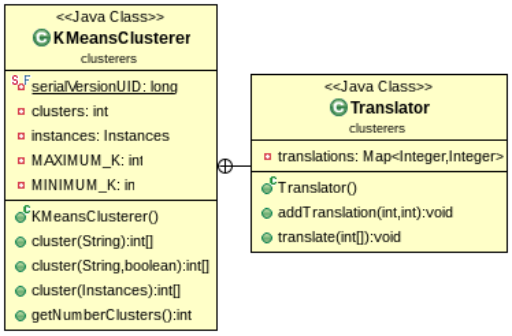

Figure 4.13: *UML* class model for the *clusterers* package content.

2. ***Configurations*** *package*. This package contains the classes used for setting user defined constants regarding the device to use, portability issues, clustering and model training fine tunning;

3. ***Controllers*** *package*. This package contains the controllers for the used, necessary to obtain and process the required hand features:

   - ***Device* interface**. Defines the mandatory behaviours for each used device;
   - ***SensoGloves* class**. Controller for the *Senso Gloves* model *DK-2*. The necessary hand features are obtained through a listener class named *SensoGlovesListener*;
   - ***LeapMotion* class**. Controller for the *Leap Motion*. The necessary hand features are obtained through a listener class named *LeapMotionListener*;
   - ***HandType* enumeration**. Enumeration used to select the hand from which to obtain the required data.

4. ***Filters* package**. This package contains the *HistogramFilter* class, which is responsible for the noise removal and compression of the cluster ID sequences produced by the clusterer module.

5. ***Gestures*** *package*. This package contains the classes that interface between the clusterers and models:

   - ***GestureDictionary* class**. Singleton class containing the implementation of the dictionary module. Uses the *KDTree* data structure provided by the *Java-ML* library and the map of *IDs* necessary for the conversion of local cluster IDs into global ones;

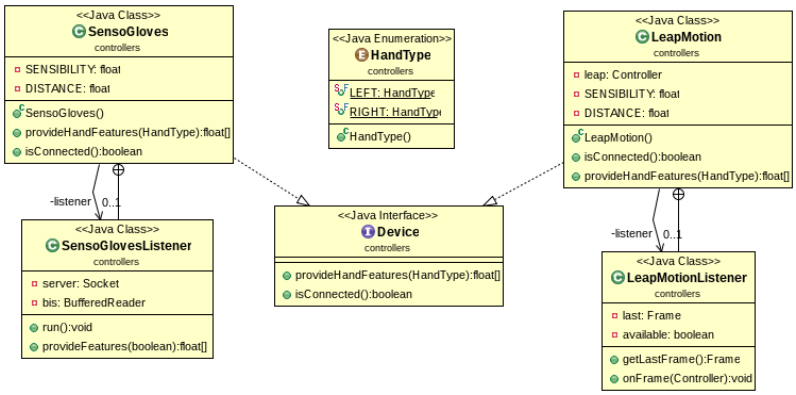

Figure 4.14: *UML* class model for the *controllers* package content.

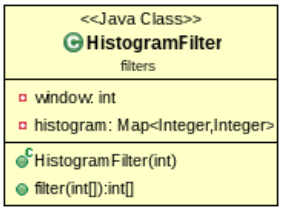

Figure 4.15: *UML* class model for the *filters* package content.

- ***DynamicGesture* class**. Every instance of this class corresponds to a distinct dynamic gesture. Eases both the clustering of the related recordings and the training of the underlying *HMM*.

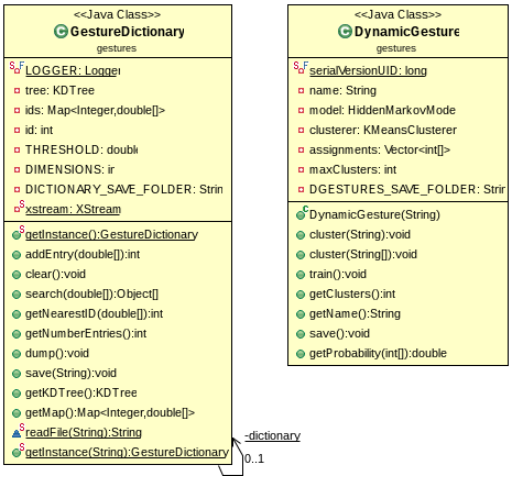

Figure 4.16: *UML* class model for the *gestures* package content.

6. ***Gui* *package***. This packages contains the following classes, necessary for the sampling and recording processes, and the only ones to require a graphical user interface:

   - ***DataSampler* class**. Implementation of the sampler module, that eases the process of sampling static left hand poses;

- ***DataRecorder* class**. Implementation of the recorder module, that eases both the recording of gesture performances and the dataset file system management. It uses a helper class *Recorder* necessary to start and stop the recording process.

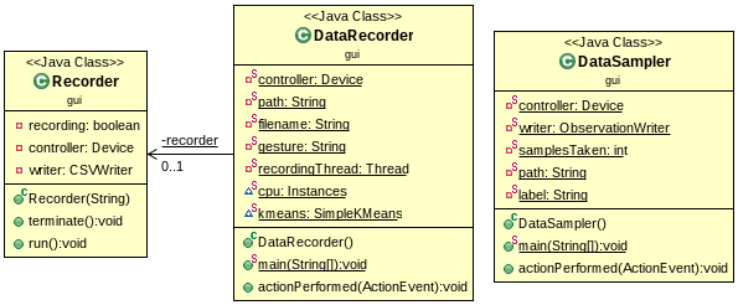

Figure 4.17: *UML* class model for the *gui* package content.

7. ***Io** package*. This package contain the following classes, necessary to the creation of files in specific formats:

   - ***ObservationWriter* class**. Used by the sampler module when writing to files the obtained static hand data. The used format is required by the *SVM* algorithms provided by the *LIBSVM* library;
   - ***CSVWriter* class**. Used by the recorder module when writing to files the dynamic gesture recordings. Outputs in the *CSV* format are simple to obtain and be used by the clusterer module.

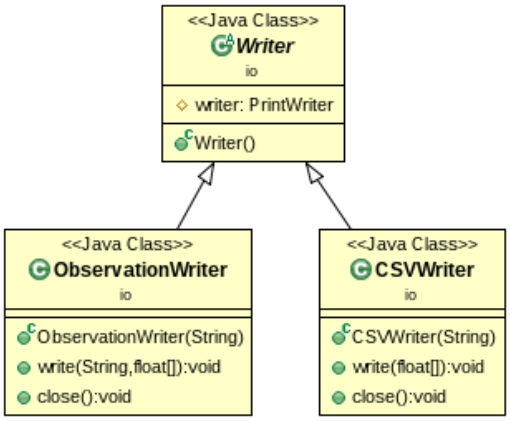

Figure 4.18: *UML* class model for the *io* package content.

8. ***Models** package*. This package contain the *Hidden Markov* class, which acts as a wrapper to *HMM* implementation provided by the *Jahmm* library, easing the model initialization, training and sequence probability calculation.

9. ***Recognizers** package*. This package contains the following classes, responsible for the gesture recognition and model training:

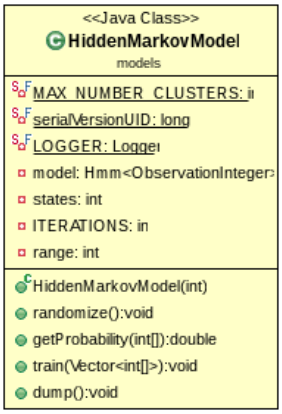

Figure 4.19: *UML* class model for the *models* package content.

- ***OfflineTrainer* class**. Implementation of both the explained pipelines for static gesture and dynamic gesture training.
- ***StaticGestureRecognizer* class**. Implementation of the explained pipeline for static gesture recognition. It uses the *State* enumeration, to keep track of the current machine state as also explained.
- ***StateMachine***. Implementation of the previously presented state machine;
- ***OnlineRecognizer* class**. Application entry point, consisting in the implementation of both the pipelines for static and dynamic gesture recognition. The mentioned circular buffer is implemented by the *CircularBuffer* class, and the hand data is obtained by a *Producer* thread and consumed by a *Consumer* thread.

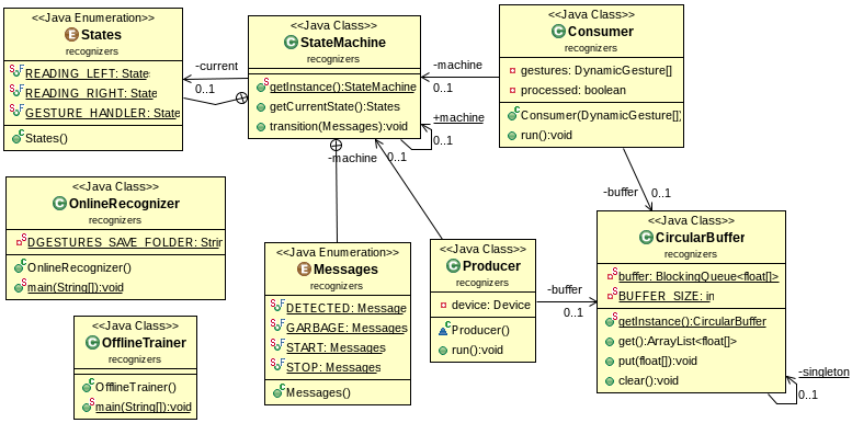

Figure 4.20: *UML* class model for the *recognizers* package content.

10. ***Validators** package*. This package contains the following classes, responsible for the implementation of the *K-Fold Cross Validation* algorithm:

- ***Fold* class**. Given a set of recordings for a particular gesture, this class contains the necessary methods to group the recordings into a training set and a testing set, according to a defined proportion.
- ***Dataset* class**. Eases the management of every fold in the gesture dataset.
- ***ConfusionMatrix* class**. Is able to provide the confusion matrixes regarding both the overall system performance and the performance of the system for each individual class.
- ***CrossValidator* class**. Implementation of the *K-Fold Cross Validation* algorithm.

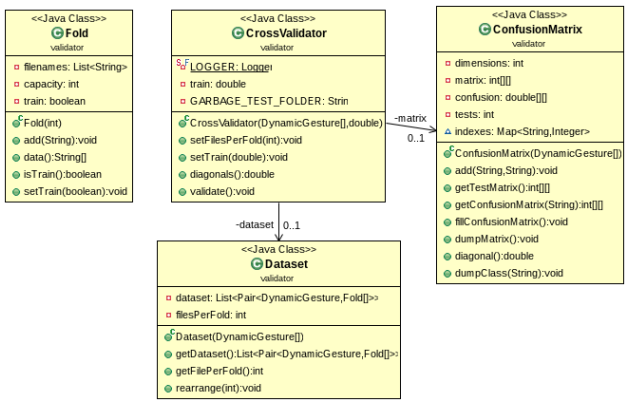

Figure 4.21: *UML* class model for the *validators* package content.

The mentioned validation algorithm is explained in the next chapter.

Chapter 5

Results

In this chapter the the experiments done and the obtained results are discussed.

5.1 Graphical User Interfaces

In this section the developed graphical user interfaces are discussed.

5.1.1 Data Sampler

The data sampler graphical user interface consists of a unique window, as depicted in figure 5.1.

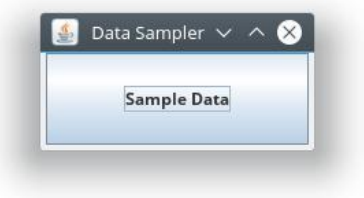

Figure 5.1: Data sampler graphical user interface.

The sampler window contains exclusively the *sampler button*. Every time the button is pressed the most recent hand data is obtained and stored in a common file, alongside the remaining data samples. When the window is closed, the sampler application stores the file properly inside the dataset directory.

5.1.2 Data Recorder

The data recorder graphical user interface also consists of a unique window, as depicted in figure 5.2.

The recorder window contains both the *start* and *stop* buttons. Every time the *start* button is pressed all the incoming data frames from the device are stored chronologically in a common

Figure 5.2: Data recorder graphical user interface.

file, until the *stop* button is pressed. When that happens, the recording application stores the file properly inside the dataset directory.

5.1.3 Gesture Prompt

In order to know which gesture is being recorded or sampled, an input form is presented to the user asking for the gesture name. This information is then used by the application to store the samples and recordings in a structured way inside the file system automating the training and recognition processes.

5.2 Results

In this section the obtained results are discussed.

5.2.1 *K-Fold Cross Validation*

In order to better evaluate the gesture recognizer module performance, a *K-Fold Cross Validation* was executed, since the number of recordings per dynamic gesture is low. By using the mentioned technique it was possible to utilize the entire volume of recordings to both train and test the recognizer module, optimizing its use and obtaining a better estimate of the overall performance.

The *K-Fold Cross Validation* algorithm initially divides the entire dataset into folds, each containing a same number of data files. Then, a given percentage of those folds are used to train the system while the remaining folds are used for testing. The algorithm iterates over the dataset, until every fold were used to both test and train the system, as depicted in figure 5.3.

For the developed system, 70% of the dataset was used for training and the remaining 30% was used for testing.

5.2.2 Confusion Matrices

Table 5.1 corresponds to the confusion matrix for the static gesture recognition module while using the *Leap Motion*. Table 5.2 is the analogous matrix using the *Senso Gloves*. Henceforward for every confusion matrix presented the rows represent the real gesture name and the columns represent the name of the recognized gesture.

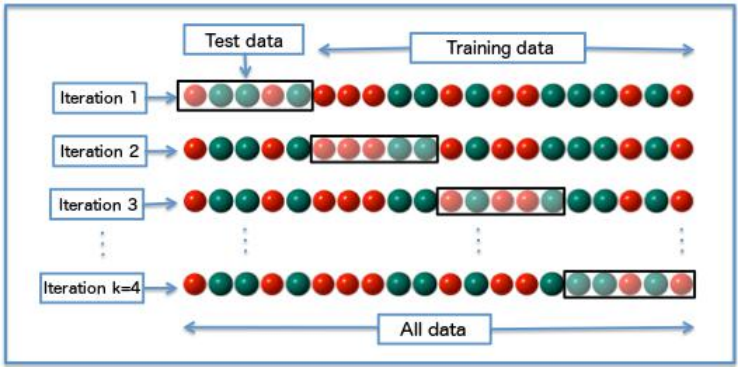

Figure 5.3: *K-Fold Cross Validation* algorithm [Joa18].

Both matrices show that using either the *Leap Motion* or the *Senso Gloves* it is possible to correctly recognize the *start* static gestures at least 60% of the time and the *stop* static gesture at least 99% of the time.

The lower recognition rates for the *start* gesture are due to the fact that the set of chosen hand features is not completely adequated to the dataset. In fact, the scatter plots for the *start* hand poses and *stop* hand pose depicted in figures 5.4 and 5.5, respectively, show that the data obtained for both gestures is to close to each other. This complicates the obtention of an ideal hyperplane while using *SVM* to train the model. However the scatter plots additionally show that the *start* hand data contains either the lowest or the greatest values per feature used, lying in the extremes of the plotted lines. This property allows to explain the 79% rate of successful classifications. The *start* recognition rate could be then improved by changing the way the hand normal, the hand velocity and the wrist normal vectors are processed. As previously stated, the components of these vectores are discretized in -1, 0 or 1 in order to increase the system robustness. Therefore, increasing the number of possible discrete values would allow a better static gesture recognition.

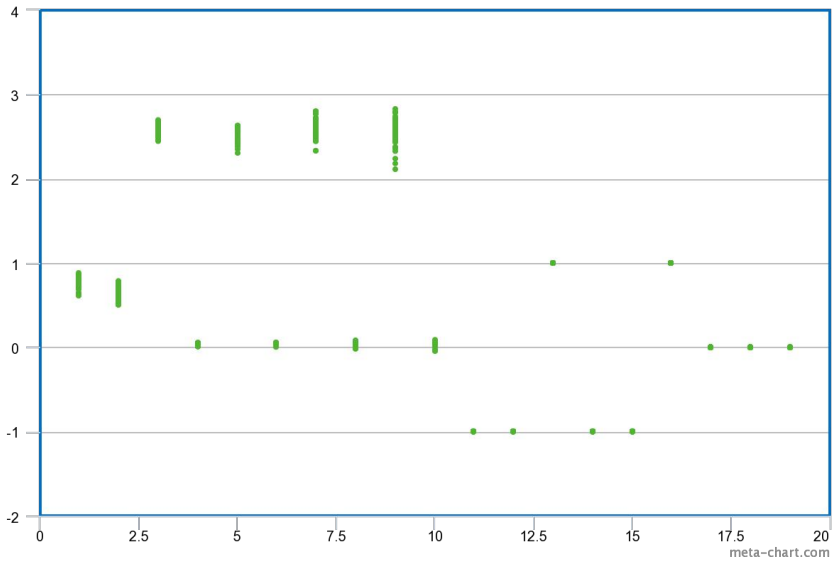

Figure 5.4: Scatter plot of the *start* hand poses obtained using the *Senso Gloves*.

	start	**stop**
start	63.0%	37.0%
stop	0.0%	100.0%

Table 5.1: Confusion matrix for the static gesture recognition module using the *Leap Motion*.

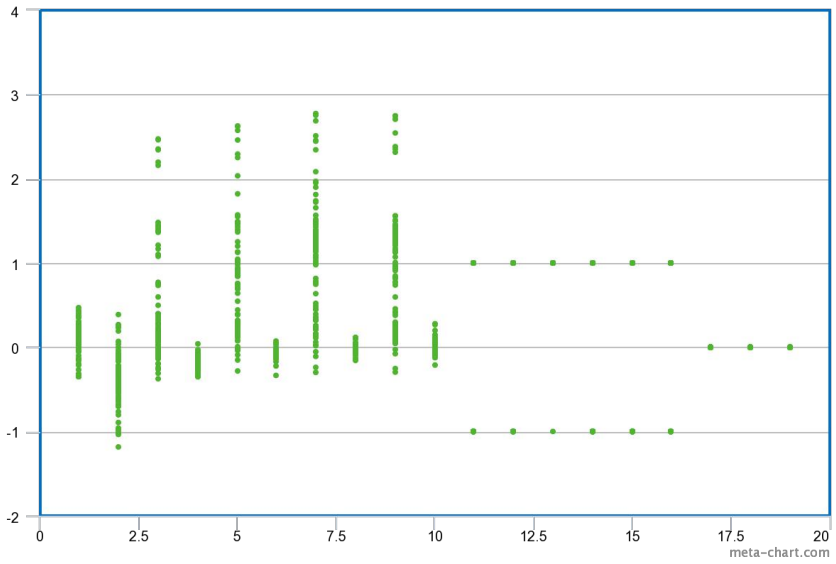

Figure 5.5: Scatter plot of the *stop* hand poses obtained using the *Senso Gloves*.

Nonetheless, since there are only two possible static gestures, the mentioned rate of correct recognitions for the *start* gesture, although not ideal, must not be perceived as alarming, since it is preferred for the robot to not do anything by default.

The difference of 16% between devices can be explained by the *Senso Gloves* greater robustness to hand occlusions and environmental noise.

Table 5.3 corresponds to the confusion matrix for the dynamic gesture recognition module while using the *Senso Gloves*. The *garbage* labelled column and row tell the proportion of times the system did neither mistook a dynamic gesture for another gesture nor could recognize the gesture within high levels of certainty.

The diagonal of the obtained matrix shows that using the *Senso Gloves* it was possible to correctly recognize every dynamic gesture at least 75% of the time, except for the *hover* gesture which was confused with the *open* gesture 33% of the times. This result can be explained given the fact that the steps required to perform an *open* gesture are similar to the steps required to perform a *hover* gesture. In fact, the first gesture can then be seen as a subset of the second, and their *HMM* must have a relevant degree of resemblance, leading to the obtained results. The results for the *hover* gesture recognition could be improved by changing the gesture, so that no gesture can be seen as a subset of another. By reducing either the mandatory distance between entries in the dictionary module or the minimum certainty level during the training phase, the variability

	start	**stop**
start	79.0%	21.0%
stop	1.0%	99.0%

Table 5.2: Confusion matrix for the static gesture recognition module using the *Senso Gloves*.

between the gestures could theoretically be improved, despite interfering with the recognition of the remaining gestures.

Table 5.4 corresponds to the confusion matrix for the dynamic gesture recognition module, while using the *Leap Motion*.

The diagonal of the this matrix shows that using the *Leap Motion* it was possible to recognize every dynamic gesture, although with lower certainty in comparison with the *Senso Gloves*, except for the *previous* gesture. This gesture was equally recognized as the *stop* gesture exactly half of the times it was not mistaken by garbage. These conflicts can also be explained with similarity issues, since both rely on the use of the index finger in circular movements that can't be correctly distinguished unless a proper tuning of the clustering algorithm properties is done, possibly compromising the recognition of the remaining gestures. A similar confusion between the *hover* gesture and the *open* gesture also occurred, but only 8% of the times. The obtained results for the *home* gesture can easily be explained by referring the limited working area of the device. Although positioned in a way that allowed to visualize the entire user torso and arms, the device showed difficulties in obtaining the correct hand pose when the same was positioned too far away from the device surface, often swapping the hand palm orientation. It should be mentioned that every recording using the *Leap Motion* was performed in a dark environment, in order to minimize the noise introduced into the system by ambient light.

Although equivalent the sets of hand features used by the *Senso Gloves* and the *Leap Motion* are distinct therefore explaining the different results obtained under the same training and testing conditions. The better results using the *Senso Gloves* can also be interpreted as a direct result of their immunity to total and partial hand occlusions, light conditions and arm extension.

	home	**hover**	**next**	**open**	**previous**	**stop**	**garbage**
home	75.0%	0.0%	0.0%	0.0%	0.0%	0.0%	25.0%
hover	0.0%	17.0%	0.0%	33.0%	0.0%	0.0%	50.0%
next	0.0%	0.0%	92.0%	0.0%	0.0%	0.0%	8.0%
open	0.0%	0.0%	0.0%	75.0%	0.0%	0.0%	25.0%
previous	0.0%	0.0%	0.0%	0.0%	92.0%	8.0%	0.0%
stop	0.0%	0.0%	0.0%	0.0%	0.0%	83.0%	17.0%
garbage	0.0%	3.0%	0.0%	0.0%	0.0%	0.0%	98.0%

Table 5.3: Confusion matrix for the dynamic gesture recognition module using the *Senso Gloves*.

	home	**hover**	**next**	**open**	**previous**	**stop**	**garbage**
home	25.0%	17.0%	0.0%	0.0%	0.0%	0.0%	58.0%
hover	0.0%	67.0%	0.0%	8.0%	0.0%	0.0%	33.0%
next	0.0%	0.0%	42.0%	0.0%	0.0%	0.0%	58.0%
open	0.0%	0.0%	0.0%	83.0%	0.0%	0.0%	17.0%
previous	0.0%	0.0%	0.0%	0.0%	25.0%	25.0%	50.0%
stop	0.0%	0.0%	0.0%	0.0%	17.0%	67.0%	17.0%
garbage	8.0%	0.0%	0.0%	0.0%	0.0%	0.0%	92.0%

Table 5.4: Confusion matrix for the dynamic gesture recognition module using the *Leap Motion*.

Chapter 6

Conclusions

In this section the project conclusions are stated and an analysis of possible future work is also discussed.

6.1 Conclusions

The following conclusions were reached based on the developed work :

- Correct recognition of both static and dynamic gestures can be achieved using the *Senso Gloves*, on average, at least 89% of the times for static gestures and at least 76% of the times for dynamic gestures. These results proved better than those obtained using the *Leap Motion*, under the same testing conditions, mainly due to the gloves' robustness to occlusions and bigger working area;
- It is possible to train a static gesture recognizer using *Support Vector Machines* with certainty levels above 79% up to 99%, while requiring a lesser data volume and computation power than those required by alternative neural network approaches, at least in cases where the number of static gestures to be recognized is small;
- Due to its ability to encode time changing gestures into sequences of finite states, *Hidden Markov Models* proved to be a good statistical model for gestural language;
- It is possible to train a dynamic gesture recognizer using the *k-Means* clustering algorithm with certainty levels ranging from 75% up to 92%. Although the use of the *Silhouette* value for optimizing the number of clusters increases the necessary time to cluster the hand data, the system was able to still recognize gestures within periods of time smaller than 1 second, since both algorithms are less computationally expensive comparatively to other clustering algorithms. The experiments made also show that the dictionary module must be well tuned, since either complete gestures or gesture subsets might appear as if clusterized in the same way, leading to severe recognition problems;

- The gestures to be recognized must be as simple, short and as distinct as possible, in order to ease the clusterization process and model training. Gestures following these criteria were better recognized. Equally the set of selected hand features must be chosen in order to increase the variability of the hand poses during the recording and live recognition phases.

6.2 Future Work

Future work may include the addition of extra dynamic gestures that would allow to either control the robot position with the hand, by means of rotating the thumb as in a gamepad joystick or to control the robotic gripper being used in a much more dedicated fashion besides the generic *open* gesture. The developed system would be more practical, at least from the human point of view, if two or more dynamic gestures could be performed in a row.

Future work may also include the implementation of other clustering algorithms in an attempt to obtain better results, namely density based clustering algorithms such as the *DBSCAN* algorithm. By properly weighting the hand features, even better results could be obtained.

The worker's safety should never be set aside, even in the most utopic human-robot collaborative scenario. Using the *Senso Gloves* haptic feedback system, a haptic mechanism for human safety could be developed, warning the human worker that his hands are to close to the robotic arm. Such system could be implemented using the absolute position tracking capabilities of the *Senso Gloves* along with *3D* sensors positioned in the workspace.

References

[Bar16] E. Barsoum. Articulated Hand Pose Estimation Review. *ArXiv e-prints*, 2016.

[CCH05] Wen-Yan Chang, Chu-Song Chen, and Yi-Ping Hung. Appearance-guided particle filtering for articulated hand tracking. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 235–242 vol. 1, June 2005.

[Col] Alex Colgan. How does the leap motion controller work?

[CPCC16] Pasquale Coscia, Francesco A. N. Palmieri, Francesco Castaldo, and Alberto Cavallo. *3-D Hand Pose Estimation from Kinect's Point Cloud Using Appearance Matching*, pages 37–45. Springer International Publishing, Cham, 2016.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.

[dCM06] T. E. de Campos and D. W. Murray. Regression-based hand pose estimation from multiple cameras. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 782–789, 2006.

[dGA16] M. M. A. de Graaf and S. Ben Allouch. Anticipating our future robot society: The evaluation of future robot applications from a user's perspective. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 755–762, 2016.

[DSD08] L. Dipietro, A. M. Sabatini, and P. Dario. A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(4):461–482, July 2008.

[DWÖG17] Endri Dibra, Thomas Wolf, A. Cengiz Öztireli, and Markus H. Gross. How to refine 3d hand pose estimation from unlabelled depth data ? In *Fifth International Conference on 3D Vision (3DV), Qingdao, China, October 10-12, 2017*, 2017.

[DYZ+17] X. Deng, S. Yang, Y. Zhang, P. Tan, L. Chang, and H. Wang. Hand3D: Hand Pose Estimation using 3D Neural Network. *ArXiv e-prints*, 2017.

[FBS15] Marco Faber, Jennifer Bützler, and Christopher M. Schlick. Human-robot Cooperation in Future Production Systems: Analysis of Requirements for Designing an Ergonomic Work System. *Procedia Manufacturing*, 3(Supplement C):510 – 517, 2015.

REFERENCES

[GVD+16] Archana Ghotkar, Pujashree Vidap, Kshitish Deo, Carole A. Vogler, Dimitri N. Metaxas, and Shanablehand K. Assaleh. Dynamic hand gesture recognition using hidden markov model by microsoft kinect sensor. 2016.

[HRH08] S. Hussmann, T. Ringbeck, and B. Hagebeuker. *A Performance Review of 3D TOF Vision Systems in Comparison to Stereo Vision Systems*, pages 103 – 20. Rijeka, Croatia, 2008.

[HSKMG09] H. Hamer, K. Schindler, E. Koller-Meier, and L. V. Gool. Tracking a hand manipulating an object. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1475–1482, 2009.

[Int17] Intel® RealSense™ Depth Camera D400-Series Datasheet . Technical report, Intel, 08 2017.

[IOA11] Nikolaos Kyriazis Iason Oikonomidis and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.101.

[ISO17] ISO. ISO 10218-1:2011 - robots and robotic devices – safety requirements for industrial robots – part 1: Robots. Available at https://www.iso.org/standard/51330.html, December 2017.

[JCP+10] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S. K. Park, M. Kim, and J. W. Jeon. FPGA design and implementation of a real-time stereo vision system. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(1):15–26, Jan 2010.

[JM00] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[Joa18] By Joan.domenech91. K-fold cross validation, 2018. [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], from Wikimedia Commons.

[Kar05] Maria Karam. A taxonomy of gestures in human computer interaction. 2005.

[Kic16] Supongkiba Kichu. Viterbi Algorithm. http://https://pt.slideshare.net/skjo2/viterbi-algorithm/, 2016. [Online; accessed 6-June-2018].

[KSGV+09] Jong-Hwan Kim, Shuzhi Sam Ge, Prahlad Vadakkepat, Norbert Jesse, Abdullah Al-Mamun, Sadasivan Puthusserypady, Ulrich Rückert, Joaquin Sitte, Ulf Witkowski, Ryohei Nakatsu, Thomas Braunl, and Jacky Baltes. Advances in robotics, fira roboworld congress 2009, incheon, korea, august 16-20, 2009. proceedings. 5744, 01 2009.

[Li14] Larry Li. Time-of-Flight Camera – An Introduction. Technical report, Texas Instruments, 05 2014.

[MF13] Aaron Martinez and Enrique Fernandez. *Learning ROS for Robotics Programming*. Packt Publishing, 2013.

REFERENCES

[MN06] Zhenyao Mo and U. Neumann. Real-time hand pose recognition using low-resolution depth images. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1499–1505, 2006.

[MYS+97] Byung-Woo Min, Ho-Sub Yoon, Jung Soh, Yun-Mo Yang, and Toshiaki Ejima. Hand gesture recognition using hidden markov models. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 5, pages 4232–4235 vol.5, 1997.

[NBT07] Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1):52 – 73, 2007.

[PKA07] David Page, Andreas Koschan, and Mongi Abidi. Methodologies and techniques for reverse engineering–the potential for automation with 3-d laser scanners, 10 2007.

[PN09] J.S. Prasad and G.C. Nandi. Clustering method evaluation for hidden markov model based real-time gesture recognition. pages 419 – 23, 2009.

[PYK+12] Sangheon Park, Sunjin Yu, Joongrock Kim, Sungjin Kim, and Sangyoun Lee. 3d hand tracking using kalman filter in depth space. *EURASIP Journal on Advances in Signal Processing*, 2012(1):36, Feb 2012.

[Rab89] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE, Volume: 77, Issue: 2*, pages 257–286, 1989.

[Rou87] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53 – 65, 1987.

[Sit] Yes, industry 5.0 is already on the horizon. http://www.machinedesign.com/industrial-automation/yes-industry-50-already-horizon. Accessed: 2018-01-30.

[SMZ+16] Srinath Sridhar, Franziska Mueller, Michael Zollhoefer, Dan Casas, Antti Oulasvirta, and Christian Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *Proceedings of European Conference on Computer Vision (ECCV)*, October 2016.

[SPM10] J.J. Sorribes, M. Prats, and A. Morales. Visual tracking of a jaw gripper based on articulated 3d models for grasping. pages 2302 – 7, Piscataway, NJ, USA, 2010.

[TJN08] Leila Takayama, Wendy Ju, and Clifford Nass. Beyond dirty, dangerous and dull: What everyday people think robots should do. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, HRI '08, pages 25–32, New York, NY, USA, 2008. ACM.

[TKD+15] Tippaya Thinsungnoen, Nuntawut Kaoungku, Pongsakorn Durongdumronchai, Kittisak Kerdprasop, and Nittaya Kerdprasop. The clustering validity with silhouette and sum of squared errors. 2015.

[TT11] Velmurugan T and Santhanam T. A survey of partition based clustering algorithms in data mining: An experimental approach. 10, 03 2011.

REFERENCES

[uni] Unimate - the first industrial robot. https://www.robotics.org/joseph-engelberger/unimate.cfm. Accessed: 2017-12-31.

[VM98] R. J. Valkenburg and A. M. McIvor. Accurate 3d measurement using a structured light system. *Image and Vision Computing*, 16(2):99 – 110, 1998.

[VMSLB13] Fereydoon Vafaei, Brian M Slator, Juan Li, and Benjamin Braaten. Taxonomy of gestures in human computer interaction, 12 2013.

[WHX17] Lin Wang, Jinfeng He, and Songjie Xu. The application of industry 4.0 in customized furniture manufacturing industry. volume 100, pages 03022 (4 pp.) –, 2017.

[Wik17] WikiBooks. Support Vector Machines. https://en.wikibooks.org/wiki/Support_Vector_Machines, 2017. [Online; accessed 6-June-2018].

[WMC+15] Xingyu Wu, Xia Mao, Lijiang Chen, Yuli Xue, and Alberto Rovetta. Depth image-based hand tracking in complex scene. *Optik - International Journal for Light and Electron Optics*, 126(20):2757 – 2763, 2015.

[WP09] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 63:1–63:8. ACM, 2009.

[WPVY17] C. Wan, T. Probst, L. Van Gool, and A. Yao. Dense 3D Regression for Hand Pose Estimation. *ArXiv e-prints*, 2017.

[YYGK17] S. Yuan, Q. Ye, G. Garcia-Hernando, and T.-K. Kim. The 2017 Hands in the Million Challenge on 3D Hand Pose Estimation. *ArXiv e-prints*, 2017.

[YYS+17] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim. BigHand2.2M Benchmark: Hand Pose Dataset and State of the Art Analysis. *ArXiv e-prints*, April 2017.

[ZLZ15] Keliang Zhou, Taigang Liu, and Lifeng Zhou. Industry 4.0: towards future industrial opportunities and challenges. pages 2147 – 52, 2015.